Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Gaber Marušič

Razvoj vtičnika Smart Plugin za programsko rešitev HP Operations Manager

diplomsko delo na univerzitetnem študiju

Ljubljana, 2016

Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Gaber Marušič

Razvoj vtičnika Smart Plugin za programsko rešitev HP Operations Manager

diplomsko delo na univerzitetnem študiju

Mentor: prof. dr. Marko Bajec

Ljubljana, 2016

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Napake v delovanju poslovnih informacijskih in tehničnih sistemov predstavljajo visoka tveganja in potencialno velike finančne izgube. S pametnim nadzorovanjem omenjenih sistemov lahko napake preprečimo preden se zgodijo. V okviru diplomske naloge opišite postopek nadzorovanja poslovnih informacijskih in tehničnih sistemov, preglejte ponudnike obstoječih rešitev ter njihove razlike. Podrobneje predstavite rešitev HP Operations Manager ter prikažite postopek razvoja razširitve za potrebe nadzora lastne aplikacije.

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Gaber Marušič, z vpisno številko **63080168**, sem avtor diplomskega dela z naslovom:

Razvoj vtičnika Smart Plugin za programsko rešitev HP Operations Manager

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom prof. dr. Marka Bajca,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 30. avgusta 2016 Podpis avtorja:

Zahvaljujem se mentorju prof. dr. Marku Bajcu za pomoč pri izdelavi diplomske naloge.

Družini

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
IT	information technology	informacijska tehnologija
FTP	file transfer protocol	protokol za prenos datotek
SOAP	simple object access protocol	protokol za komunikacijo med spletnimi storitvami
RPC	remote procedure call	oddaljeni klic podprograma
BBC	communication broker	komunikacijski posrednik
ACL	access control list	seznam nadzora dostopa
PAM	pluggable authentication module	vtični overitveni modul
SNMP	simple network management protocol	preprost protokol upravljanja z omrežji
HDD	hard disk drive	trdi disk
SSD	solid state drive	polprevodniški disk
VHD	virtual hard disk	virtualni trdi disk
S.M.A.R.T	self-monitoring, analysis and reporting tehnology	tehnologija samonadziranja, analiziranja in poročanja
XML	extensible markup language	razširljiv označevalni jezik
XSD	XML schema definition	definicija sheme XML
RAID	redundant array of independent disks	redundantno polje samostojnih diskov
SATA	serial AT attachment	serijski AT priklop
SCSI	small computer system interface	vmesnik za majhen računalniški priklop

Povzetek

Naslov: Razvoj vtičnika Smart Plugin za programsko rešitev HP Operations Manager

Razvoj in napredek informacijskih tehnologij, povečuje zapletenost in zahtevnost poslovnih in podpornih sistemov v organizacijah in podjetjih. Napake v delovanju, predstavljajo vse večje tveganje in poslovne izgube. Pojavila se je težnja, po nadzoru strateško pomembnih sistemov z vpeljavo programskih rešitev za nadzorovanje. V diplomskem delu je opisano delovanje programske opreme za nadzorovanje HP Operations Manager. Zaradi želje po nadzorovanju lastnih aplikacij, je prikazan postopek razvoja vtičnika Smart Plugin, ki razširja področja delovanja HP Operations Manager. Razvoj vtičnika zajema izvedbo agenta za pridobivanje podatkov iz aplikacije za upravljanje diskovnih pomnilnikov in konfiguracije HP Operations Manager, ki služi pošiljanju sporočil in obveščanju operaterja. Operater ima pregled nad delovanjem sistema, spremlja sporočila in z uporabo orodij ter akcij preprečuje in odpravlja napake.

Ključne besede: nadzorovanje, sporočila, dogodki, IT infrastruktura, HP Operations Manager, Smart Plugin.

Abstract

Title: Development of Smart Plugin for HP Operations Manager software

Development and progress of informational technologies, increases complexity of business and support systems in enterprises. Errors in operation, can cause great risks and potential business losses. There is a need for applications which monitors critical strategic systems. The thesis discusses concepts of HP Operations Manager monitoring software. It describes development process of Smart Plugin for HP Operations Manager, which extends monitoring capabilities to custom made application for managing disk storage systems. Development includes implementation of data collection agent which gathers information from monitored system and HP Operations Manager configuration used for sending messages to operator. Operator has an overview of system's current state and detects, prevents and solves problems using appropriate tools and corrective actions.

Keywords: monitoring, messages, events, IT infrastructure, HP Operations Manager, Smart Plugin.

Poglavje 1

Uvod

Z razvojem informacijskih tehnologij v poslovnih in podpornih procesih organizacij in podjetij, postajajo informacijski sistemi čedalje bolj zapleteni in zahtevni. Nedelovanje oziroma nepravilno delovanje le-teh, lahko predstavlja poslovno izgubo ter nepravilnosti pri poslovanju in delovanju podjetja. Infrastruktura IT in poslovne aplikacije, ki jih podjetja uporabljajo, so pogosto razpršene po omrežju, ki ga sestavljajo med seboj povezani, heterogeni in pogosto geografsko oddaljeni strežniki. Prednosti decentraliziranega pristopa so izogib eni točki odpovedi infrastrukture, razbremenitev glavnega omrežja in strežnika, ter prenos nadzora in upravljanja v operativni oddelek podjetja. Medsebojna povezanost strežnikov prinese večjo prilagodljivost, saj so informacije dostopne uporabnikom ne glede na to, kje se nahajajo. Decentralizirano oziroma razpršeno infrastrukturo, je ravno zaradi njene prilagodljivosti težje nadzorovati in upravljati. Strežniki se lahko med seboj razlikujejo po strojni opremi, operacijskih sistemih ali verziji nameščene programske opreme; uporabniki pa pogosto prilagajajo aplikacije svojim potrebam. Zaradi teh razlogov je težko zaznati ali preprečiti dogodke, ki privedejo do napak. Pojavi se potreba po pametnem sistemu za nadzorovanje infrastrukture IT in aplikacij informacijskih sistemov. Z vpeljavo sistema za nadzor, spremljamo vsakodnevne naloge, ki se nanašajo na upravljanje posameznih komponent infrastrukture IT, omrežij, storitev in individualnih ter poslovnih

aplikacij. V primeru uporabe specifične aplikacije drugih razvijalcev oziroma lastne aplikacije, pa nam sistemi za nadzorovanje po večini nudijo možnost lastnih integracij; vtičnikov Smart Plugin oziroma programskih paketov Management Pack v naše okolje. Te skrbijo za pridobivanje podatkov in nadzorovanje tovrstnih aplikacij. Z njimi samodejno zaznavamo, prioritiziramo in analiziramo incidente, ki se dogodijo ter zmanjšamo čas, ki je potreben za njihovo odpravo. S tem posledično razbremenimo osebje IT monotonih vsakodnevnih opravil in jim lahko dodelimo strateško pomembnejše naloge. Za podjetja so prednosti uporabe tovrstnih sistemov zmanjšanje časa izpada in pridobivanje informacij, poročil ter analitik v realnem času. Zmanjšujejo stroške in večajo učinkovitost operativnega upravljanja infrastrukture IT na račun pametnega obvladovanja večjih količin podatkov. V diplomskem delu so opisani glavni koncepti delovanja sistemov za nadzor delovanja IT in predstavitev ene tovrstne komercialne aplikacije. Prikazan je razvoj vtičnika za nadzorovanje Smart Plugin in njegova integracija v aplikacijo HP Operations Manager.

Poglavje 2

Nadzorovanje aplikacij in poslovnih okolij

2.1 Sistemi za nadzorovanje

Nadzorovanje je izraz, čigar pomen je močno odvisen od konteksta. V širšem pogledu, se nanaša na proces zavedanja stanja sistema v danem trenutku. To lahko dosežemo na dva načina. Proaktivno in retroaktivno. Proaktivno vključuje spremljanje vizualnih indikatorjev, časovnic in grafov, kar po navadi administratorji imenujejo nadzorovanje, retroaktivno pa vključuje samodejno dostavo obvestil operatorju, ki pritegnejo njegovo pozornost ob večji spremembi stanja sistema. To označujemo z izrazom obveščanje [7]. Tehnike uporabljene pri nadzoru informacijskih sistemov vključujejo področja procesiranja statističnih in podatkovnih analiz v realnem času. Programske komponente, ki so uporabljene pri pridobivanju podatkov, njihovem procesiranju in predstavitvi, imenujemo sistemi za nadzor. Obveščanje je zmožnost, da zaznajo in obvestijo operatorja o dogodkih, ki nakazujejo na spremembo v sistemu. Obvestilo je lahko preprosto sporočilo (SMS, elektronska pošta, takojšnje sporočilo) ali pa se zabeleži, v sistemu za sledenje težavam. Sistem za nadzor je skupek programskih komponent, ki merijo, zbirajo in interpretirajo podatke. Tak sistem je optimiziran za učinkovito hranjenje in vizualno

predstavo nadzorovanih metrik. Na tržišču je veliko komercialnih in odprtokodnih rešitev, ki se med seboj razlikujejo po namenu in funkcionalnostih, ki jih ponujajo. Preden se odločimo za katero izmed njih, je dobro, če se vprašamo naslednje:

- Kakšni so stroški vpeljave sistema za nadzor?
- Je rešitev dovolj robustna in stabilna, da ne bomo porabili preveč časa za upravljanje le-te?
- Nam omogoča nadzorovanje v zadosti majhnih oziroma velikih časovnih okvirih?
- Ali ponuja uporabniku aplikacijske programske vmesnike, ki jih lahko uporabi za izdelavo lastne razširitve?
- Kako preprosto je tak sistem razširiti?

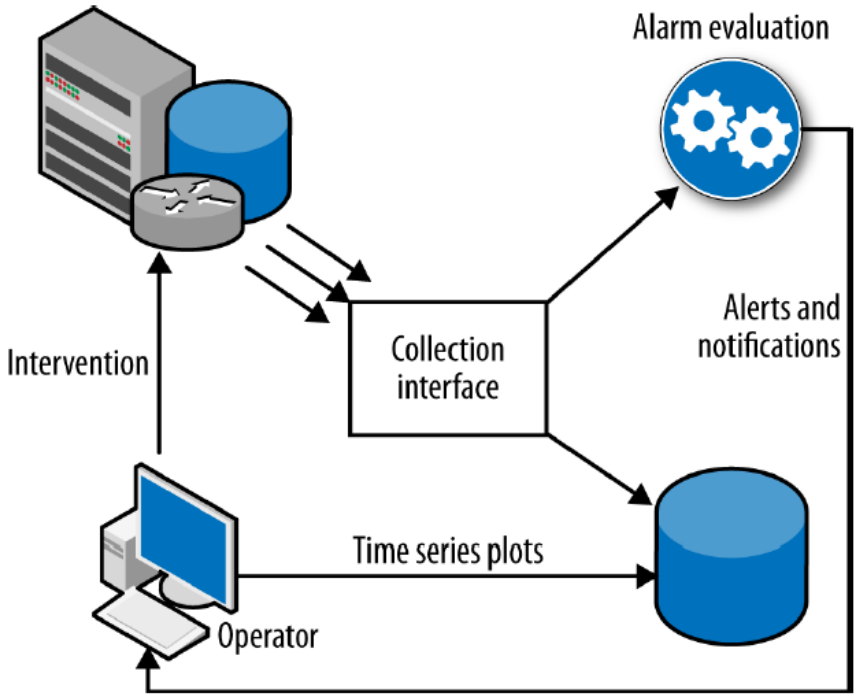
Večina sistemov ima zelo podobno visokonivojsko arhitekturo in delujejo po podobnih principih (slika 2.1). Ločimo jih po tem kako poteka komunikacija med strežnikom in agentom: ali strežnik sprašuje agenta ali agent pošilja informacije na strežnik [19]. Proces se začne s pridobivanjem vhodnih podatkov. Agent, ki je nameščen na nadzorovanem sistemu, zbere in posreduje podatke sistemu za nadzor. Ta shrani podatke in metrike in preveri ali je prišlo do prekoračitve praga katerega izmed pogojev. Kadar je prag presežen, sistem pošlje obvestilo o napaki operaterju. Naloga operaterja je, da obvestilo analizira in ustrezno ukrepa. Proces lahko v grobem razdelimo na tri dele [7].

1. Pridobivanje podatkov:
   Agent zbere podatke o delovanju sistema iz strežnikov, podatkovnih baz ter omrežne ali druge infrastrukturne opreme. Vir podatkov so dnevniške datoteke, statistike naprav in sistemske meritve. Agent združuje vhodne podatke v metrike in jih opremi z meta informacijami. Nato

jih posreduje sistemu za nadzor po vnaprej določenih komunikacijskih protokolih in shrani v podatkovno bazo metrik.

2. Obdelovanje in shranjevanje podatkov:
   Komponenta obveščanja oceni, ali se v metrikah pojavljajo kakšne anomalije. Kadar sistem zazna nepravilnosti, sproži alarm in pošlje obvestilo operaterju.

3. Predstavitev:
   Operater preveri trenutno stanje in če ugotovi, da gre za napako, primerno ukrepa. Po razrešitvi napake, mora sistem sporočiti uporabniku, da je bila napaka odpravljena.

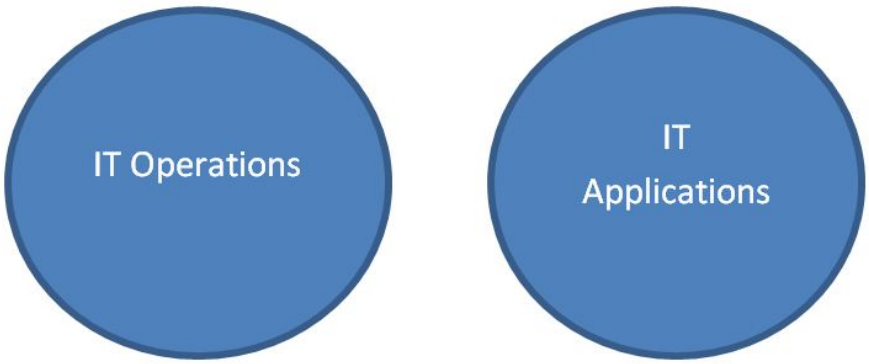


Slika 2.1: Potek delovanja sistemov za nadzor.

Sistem za nadzor ponuja enotno referenčno točko vsem operaterjem, ki igrajo ključno vlogo pri nadzorovanju. Njegove prednosti pridejo najbolj do izraza pri velikih in zrelih organizacijah, kjer lahko sistemski administratorji, skrbniki infrastrukture in razvijalci sodelujejo med seboj in izmenjujejo svoja opažanja in izkušnje. Enotna referenčna točka vseh vpletenih, znatno povečuje odzivnost in blaži resnost problemov. Take sisteme opisujemo z izrazom upravljanje delovanja IT [7].

2.2 Delovanje informacijske infrastrukture

Za potrebe razumevanja sistemov upravljanja delovanja IT, moramo najprej vedeti, kaj spada v to domeno. Na ta način se lažje odločimo, kako čimbolj učinkovito uvesti sistem za nadzorovanje, ki nam bo nudil koristne informacije. Termin delovanje IT, označuje množico procesov in storitev, ki jih osebje IT omogoča svojim notranjim ali zunanjim strankam ali pa jih uporabljajo za potrebe lastnega delovanja. Po navadi, se ne ukvarja z razvojnimi aktivnosti in programiranjem, ki spada v področje programskega IT, zato definiramo delovanje IT kot pojem, ki zavzema vse funkcije IT razen razvoja in upravljanja aplikacij. Delovanje IT in programski IT področji sta prikazani na sliki 2.2.

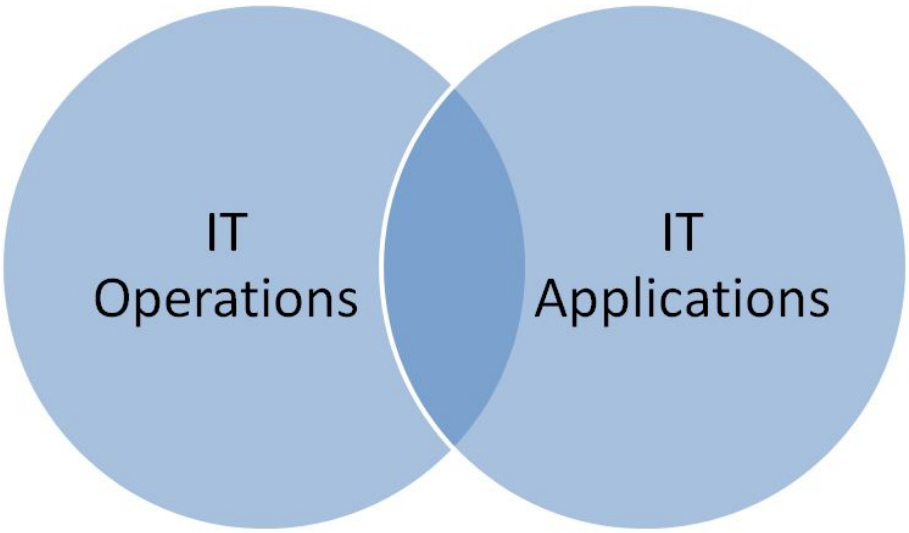


Slika 2.2: Področja IT.

Definicija velja v teoriji, vendar v praksi po navadi ne drži. Poznamo veliko področij, kjer delovanje IT in programski IT sovpadata. Ta so naslednja:

- namestitev in konfiguracija programske opreme za podporo poslovnim aplikacijam,
- vzdrževanje podatkovnih baz. Programski IT jih vzdržuje z vidika integritete podatkov, delovanje IT pa z vidika učinkovitosti, ter podpornih procesov (brisanje starih zapisov, ponovno indeksiranje, varnostno kopiranje itd.),
- odpravljanje težav aplikacij,
- nadzorovanje zmogljivosti in napak aplikacij in opozarjanje na probleme,
- konfiguracija komunikacijskih in drugih omrežij s poslovnimi partnerji, strankami in ostalimi entitetami, kot so: FTP strežniki, elektronska izmenjava podatkov med organizacijami, kopiranje podatkov med strežniki itd.,
- podpora uporabnikom, katerih naloge so razdeljene med programskim IT in področjem delovanja IT,
- integrirana specializirana oprema s posebnimi aplikacijami, kot so: optični čitalci, tablični računalniki, industrijski tiskalniki, mobilnimi odjemalci itd.

Posodobljen Vennov diagram prikazuje slika 2.3.



Slika 2.3: Področja IT.

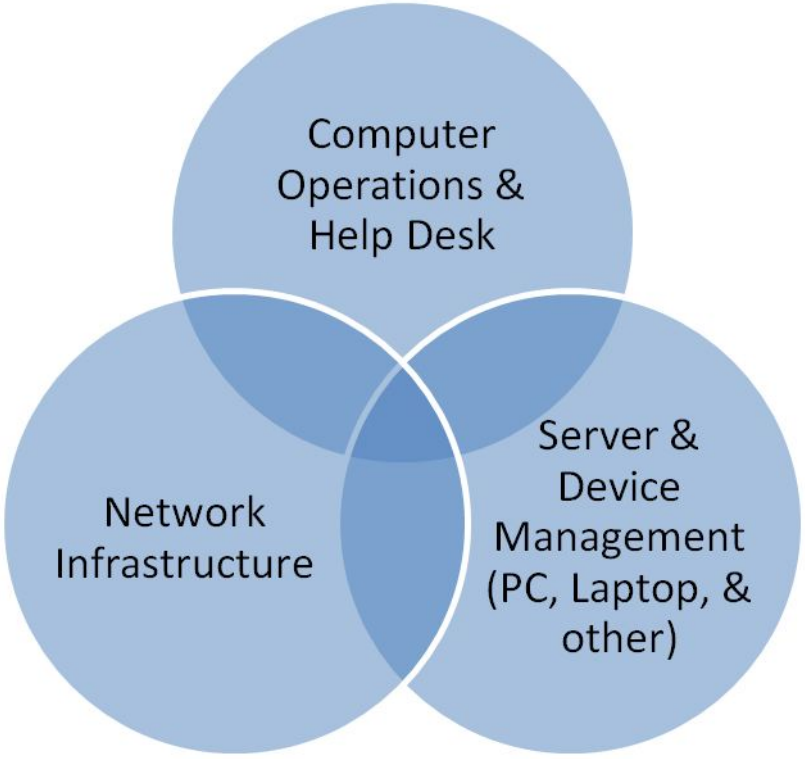
Presek označuje zgoraj našteta področja s katerimi se ukvarja delovanje IT. V kolikor želimo pokriti vsa področja, moramo dodati tudi tista, ki nimajo ničesar skupnega s programskim IT. Ločimo jih v tri glavne skupine, ki vsebujejo naslednja področja:

- Upravljanje z računalniškimi sistemi in podpora:
  - upravljanje s podatkovnimi centri,
  - varnostno kopiranje podatkov,
  - upravljanje z uporabniki,
  - podpora uporabnikom.
- Omrežna infrastruktura:
  - telekomunikacije,
  - omrežno varnost,

  - oddaljeni dostopi,
  - nadzorovanje pravilnosti delovanja omrežja ter opozarjanje na napake.

- Upravljanje s strežniki ter napravami:
  - konfiguracija,
  - vzdrževanje,
  - nadgradnje,
  - popravila.

Slika 2.4 prikazuje diagrama novega delovanja IT področja.



Slika 2.4: Področje delovanja IT.

Definicijo lahko razširimo sledeče: *"Področje delovanja IT je odgovorno za nemoteno delovanje infrastrukturnih funkcij in operativnih okolij, ki skrbijo za podporo razvoju aplikacij, notranjim ter zunanjim strankam, omrežni infrastrukturi, strežnikom ter drugim napravam in storitvam podpori uporabnikom [20]".*

2.3 Programske rešitve sistemov za nadzorovanje

2.3.1 Microsoft System Center Operations Manager

Microsoft System Center Operations Manager (v nadaljevanju SCOM) je izdelek, ki sodi v programski paket Microsoft System Center. Ker gre za Microsoftovo rešitev, je podpora sistemov UNIX in Linux okrnjena. Omogoča nadzorovanje tako operacijskih sistemov kot hipervizorjev in prikazuje njihovo stanje, zdravje in informacije o zmogljivosti. SCOM omogoča nadzor tudi brez namestitve agenta na ciljni sistem, vendar pa v tem načinu ne moremo doseči enake stopnje funkcionalnosti [9]. Njegova največja prednost je veliko število programskih paketov Management Pack, ki razširjajo nadzorovanje na vse večje ter bolj razširjene poslovne aplikacije. Ker je Microsoft ponudnik velikega števila različnih produktov, se SCOM zlahka integrira tudi z njimi. Nadzorovanje strežnika Exchange, podatkovne baze Microsoft SQL Server, Skype for Business in ostalih aplikacij deluje brez težav, saj razvijalci teh produktov tesno sodelujejo med seboj.
Glavna slabost SCOM je visoka cena, saj je zahteva licenco tako za strežnik, kot tudi za odjemalce in podatkovno bazo Microsoft SQL Server. Prav tako je SCOM slabša izbira v kolikor uporabljamo okolja UNIX in Linux.

2.3.2 Nagios

Nagios je brezplačna, odprtokodna rešitev za nadzorovanje sistemov, infrastrukture in omrežij. Prvotno je bila narejena za okolje Linux, vendar se jo

z uporabo ustreznih vtičnikov, da razširiti tudi na nadzorovanje okolij Windows. Nagios Core je nizko nivojski set orodij, ki ne vsebuje administracijske konzole in modernega grafičnega vmesnika. To lahko zaobidemo z uporabo katerega od dodatkov, ki razširjajo funkcionalnost. Na voljo je tudi plačljiva različica Nagios XI, ki že vsebuje veliko število vtičnikov in administracijsko konzolo.
Glavni problem Nagios je, da zahteva za namestitev in vzdrževanje veliko tehničnega znanja. Ima strmo krivuljo učenja, saj poteka konfiguracija ročno z uporabo konfiguracijskih datotek. Slabost Nagiosa je tudi slabo razširjanje na zelo velika okolja, saj ima težave pri odkrivanju velikih topologij omrežij, naprav in strežnikov. Kljub temu, pa je danes Nagios še vedno vodilno odprtokodno orodje za nadzorovanje, predvsem na račun velike odprtokodne skupnosti [18].

2.3.3 Zabbix

Zabbix je odprtokodna rešitev, ki omogoča nadzorovanje dosegljivosti in zmogljivosti IT infrastrukture. Omogoča pošiljanje obvestil in izvajanje oddaljenih ukazov. Deluje v načinu strežnik-agent s tem, da omogoča dva načina delovanja agenta. V pasivnem načinu strežnik poda zahtevo kaj naj agent nadzoruje, v aktivnem načinu pa agent sprašuje strežnik katere vrednosti naj mu pošlje. Zabbix podpira okolja UNIX, Linux in Windows s katerih pridobiva podatke in jih prikazuje v spletni administracijski konzoli. Odlikujejo ga enostavna uporaba in nizka poraba virov na nadzorovanih sistemih in sodobna implementacija konceptov nadzorovanja. Uporabniku ne ponuja razširitev in možnosti nadzorovanja lastnih aplikacij v taki meri, kot jih ponujajo ostale programske rešitve.
Zabbix je primarno namenjen nadzorovanju omrežnih storitev in omrežne opreme in ima okrnjeno funkcionalnost v primerjavi z ostalimi sistemi za nadzorovanje [14].

2.3.4 IBM Tivoli Monitoring

IBM Tivoli Monitoring (v nadaljevanju ITM), je orodje za nadzorovanje sistemov in infrastrukture IT. Tako kot ostala orodja s tega področja, nam omogoča krajši čas odkrivanja težav, uporabniški vmesnik za nadzor, prikazovanje podatkov in pošiljanje sporočil ob zaznanih nepravilnostih. Arhitekturo ITM sestavljajo glavni strežnik, ki zbira podatke iz agentov ter pomožnih strežnikov, strežnik, ki je namenjen dostopu uporabnikov do administracijske konzole in podatkovne baze, ki hrani podatke o metrikah in poročilih. Z razliko od ostalih rešitev ima ITM tri vrste agentov, ki so namenjeni specifični uporabi. Agent za nadzor operacijskih sistemov nudi osnovno nadzorovanje računalniških sistemov, aplikacijski agent s katerim nadzorujemo specifične aplikacije, kot so podatkovne baze, strežnik Exchange in .Net aplikacije in univerzalni agent, ki je popolnoma nastavljiv in pridobiva podatke iz različnih virov. Omogoča nam nadzorovanje lastnih aplikacij in integracijo v ITM. Kljub temu, da ITM uporablja enoten uporabniški vmesnik, si ga lahko uporabniki nastavijo po svojih željah, saj omogoča prilagodljivost glede na vrsto uporabnika. ITM nudi široko paleto podprtih okolij, prav tako pa se zlahka integrira v ostale rešitve IBM Tivoli programskega paketa.
Prednosti ITM so takojšnje delovanje po namestitvi. Ponuja veliko privzetih nadzornih plošč in ne zahteva veliko znanja od uporabnika. V novejši različici ITM uporablja SOAP za komunikacijo med strežnikom in agenti, kar pomeni da pri proženju dogodkov nismo omejeni s programskim jezikom, saj SOAP deluje z večino izmed njih. ITM je trenutno vodilna programska oprema za nadziranje sistemov in infrastrukture IT [13].

2.3.5 HP Operations Manager

HP Operations Manager (v nadaljevanju HPOM) je komercialno orodje podjetja HP, zgrajeno na temeljih orodja HP Network Node Managerja, ki se je uporabljalo za nadzor omrežij. Omogoča nadzorovanje različnih vidikov sistemov od infrastrukture do aplikacij. Deluje tako, da agent, ki je nameščen na

nadzorovan sistem pošilja podatke centralnemu strežniku (ang. agent-push). HPOM razvijalcem nudi aplikacijske programske vmesnike, ki jih lahko uporabimo v lastnih programih in s tem pošiljamo sporočila neposredno agentu. HPOM podpira okolja Linux, UNIX in Windows in vsebuje spletno in namizno administracijsko konzolo, preko katere lahko nadzorujemo naše okolje. Funkcionalnost HPOM lahko razširimo z namestitvijo vtičnikov Smart Plugin (v nadaljevanju SPI), ki omogočajo nadzorovanje tudi drugih aplikacij in sistemov [12].

Principi delovanja vseh treh produktov so zelo podobni, vendar po mojem mnenju HPOM združuje dobre lastnosti obeh svojih tekmecev. Ponuja dobro razširljivost z vtičniki SPI, enostavno uporabo ter uporabniško administracijsko konzolo, ki so prednosti SCOM, po drugi strani pa podporo različnih okolij, ter robustnost delovanja, ki jo nudi Nagios. Primerjavo naštetih sistemov za nadzorovanje prikazuje tabela 2.1. V diplomskem delu sem opisal delovanje HPOM ter prikazal razvoj vtičnika SPI na primeru aplikacije za upravljanje diskovnih pomnilnikov (v nadaljevanju GDM) v velikem poslovnem okolju.

	SCOM	**Nagios**	**Zabbix**	**ITM**	**HPOM**
Integracija	Microsoft produkti	Vtičniki tretjih strank	Ne	IBM produkti, Microsoft produkti	HP produkti
Nadzorovanje aplikacij	Z Management Pack	Z vtičniki	Ne	Z univerzalnim agentom	S SPI
Obveščanje	Da	Da	Da	Da	Da
Odkrivanje	Da	Da	Z vtičniki	Da	Da
Podpora okolij	Windows, UNIX	UNIX, Linux	UNIX, Linux, Windows	UNIX, Linux, Windows	UNIX, Linux, Windows
Vrsta nadziranja	Z ali brez agenta	Z agentom	Z agentom	Z ali brez agenta	Z agentom
Konfiguracija	Konzola	Datoteke	Konzola	Konzola	Konzola in skripti
Cena	Zelo visoka	Zastonj	Zastonj	Visoka	Srednja
Poročila	Da	Da	Ne	Da	Da

Tabela 2.1: Primerjava programskih rešitev za nadzorovanje.

Poglavje 3

Programska oprema HP Operations Manager

HPOM je porazdeljena programska rešitev, ki deluje v načinu odjemalec-strežnik. Namenjena je sistemski administratorjem kot orodje, ki zazna, rešuje in preprečuje probleme, ki nastanejo na področju IT delovanja v podjetju. HPOM je stopnjevalno razširljiva programska rešitev. Omogoča visoko mero nastavljivosti in ustreza velikemu spektru potreb informacijskih tehnologij organizacije ter njenih uporabnikov. Sistemski administratorji lahko razširijo področje uporabe HPOM z integracijo upravljavskih aplikacij drugih partnerjev in razvijalcev.
Prednosti, ki jih vpeljava HPOM prinese:

- preprečevanje problemov z uporabo preventivnih akcij,
- zmanjševanje časa izpada končnih uporabnikov zaradi nedelovanje sistema,
- maksimiranje dosegljivosti in delovanja omrežnih komponent,
- razbremenitev administratorjev z avtomatskim reševanjem problemov,
- zmanjševanje časa reševanja problemov,

- zmanjševanje stroškov upravljanja okolij.

HPOM proaktivno pomaga zaznavati in reševati probleme, ki lahko nastanejo v distribuiranih heterogenih okoljih, ki vsebujejo omrežne komponente, sisteme, ter aplikacije. Nudi tudi mehanizem za dokumentiranje rešitev, katerega uporabimo kadar slutimo, da bomo naleteli na podobne težave v prihodnosti [3].

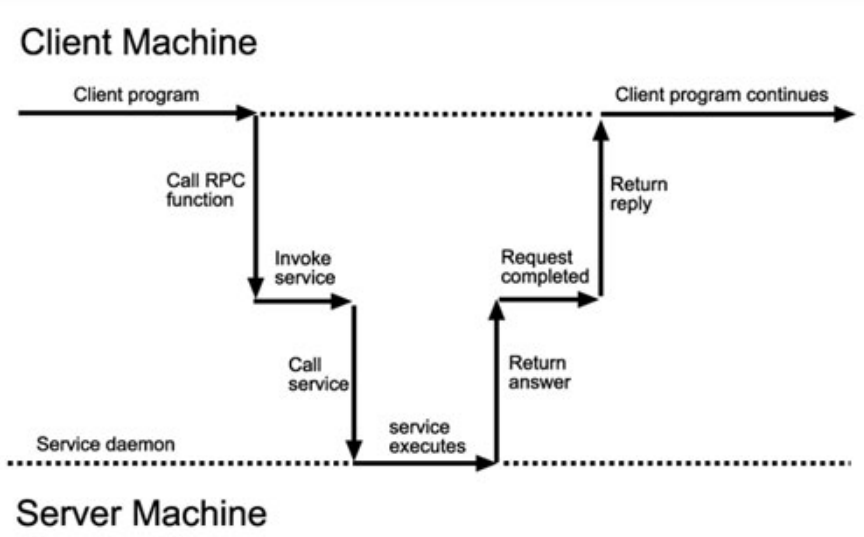
3.1 Koncept delovanja odjemalec-strežnik

Koncept delovanja HPOM temelji na komunikaciji med upravljavskim strežnikom ter upravljanimi vozlišči. HPOM in agent HP Operations Agent (v nadaljevanju OA) komunicirata s pomočjo oddaljenih klicev podprogramov RPC, ki temeljijo na BBC, signalih, vrstah in ceveh. Ta mehanizem velja za komunikacijo med strežnikom in upravljanimi vozlišči, in med procesi, ki tečejo lokalno na strežniku. OA zbira podatke o dogodkih na upravljanem vozlišču, ter posreduje pomembne informacije upravljavskemu strežniku [1].

3.2 Oddaljeni klici podprogramov

V porazdeljenih računalniških okoljih je klic RPC, ko program, v našem primeru komponenta HPOM, sproži proceduro, ki naj se izvrši na drugem naslovnem prostoru. To je tipično drug računalnik v omrežju, lahko pa se izvrši tudi lokalno. Procedura RPC je implementirana brez eksplicitnih podrobnosti o oddaljeni interakciji in se obnaša enako ali je zagnana lokalno ali oddaljeno. RPC je tip zahteva-odgovor protokola. Odjemalec sproži RPC, ki pošlje sporočilo znanemu strežniku z zahtevo o izvedbi procedure s priloženimi parametri. Oddaljeni strežnik pošlje odgovor odjemalcu. Potek klica RPC je prikazan na sliki 3.1. V kolikor ne uporabljamo asinhronih klicev RPC, je odjemalec zaustavljen dokler strežnik ne obdela zahteve. Obstaja veliko različnih implementacij, zaradi česar se tudi protokoli RPC med seboj razlikujejo. Pomembna razlika med oddaljenimi klici in lokalnimi je

ta, da so oddaljeni klici podvrženi napakam v omrežju in komunikacijskim napakam [15].



Slika 3.1: Oddaljen klic podprograma.

3.3 Varnost

Varnost v sistemu HPOM vključuje več kot le pravilno konfiguracijo. Poskrbeti moramo za varnost v celotnem obsegu okolja, ki ga nadziramo. Varnost delimo na naslednja področja:

- sistemska varnost, ki omogoča, da upravljavski strežnik HPOM in upravljano vozlišče delujeta kot zaupanja vreden sistem,
- omrežna varnost, ki ščiti podatke, ki se izmenjujejo med upravljavskim strežnikom in upravljanim vozliščem.
- varnost znotraj HPOM, ki skrbi za varnostne aspekte aplikacij in njihovega izvajanja, operaterskih akcij in konfiguracije HPOM.

3.3.1 Sistemska varnost

Pred namestitvijo programske opreme HPOM moramo zagotoviti varnost na sistemski ravni. V poslovnih okoljih nam to po navadi določajo varnostna pravila podjetja in industrijski standardi, katerim moramo ustrezati. HPOM mora biti nastavljen tako, da te standarde podpira in jih uporablja.

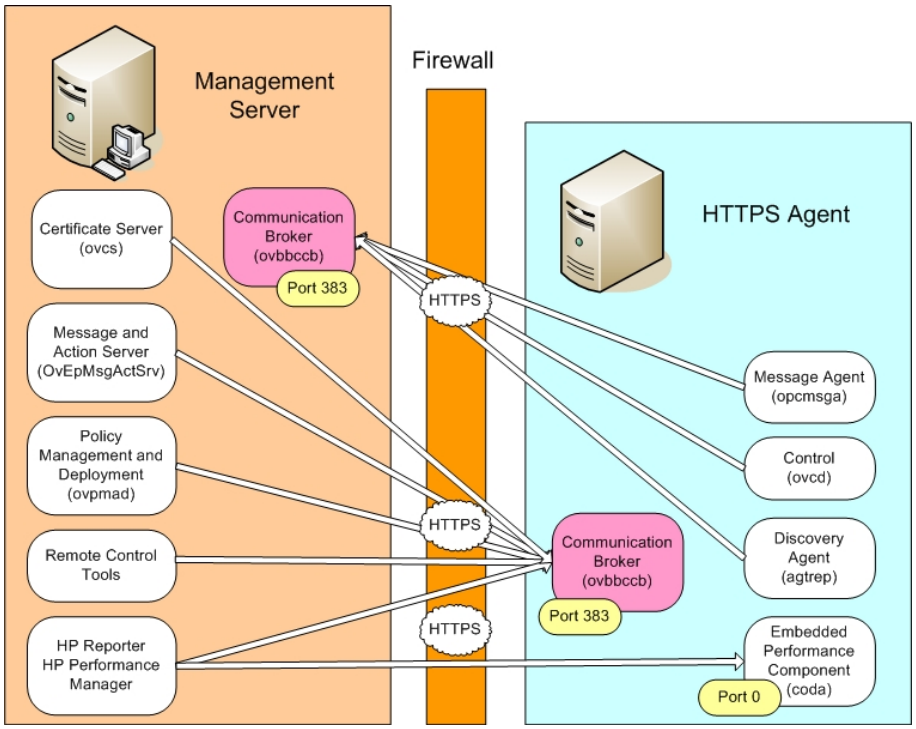
- Overitev:
  Varnostna pravila definirajo stroga določila glede uporabniških gesel ter metod overjanj. HPOM podpira PAM pri overjanju uporabnikov pri dostopu do administracijske konzole.

- Oddaljen in terminalni dostop:
  Sistemska varnostna politika lahko omejuje dostop do terminalov in oddaljen dostop z naduporabnikom. V tem primeru moramo opraviti namestitev OA na oddaljen sistem ročno.

- Dostop do datotek:
  Nekatera varnostna pravila priporočajo uporabo ACL, ki so seznami dovoljenj kateri uporabniki lahko uporabljajo določene datoteke in kakšne operacije lahko nad njimi izvajajo. HPOM ne uporablja ACL, vendar uveljavlja stroge omejitve dostopa in varuje pomembne datoteke s šifriranjem ali uporabo digitalnih podpisov.

3.3.2 Omrežna varnost

V HPOM je poudarek na varnosti povezav med procesi. Varne povezave so lahko znotraj omrežja, med več različnimi omrežji ali preko požarnih zidov. HPOM omogoča robustno varnost ne glede na to ali je strežnik ali omrežje upravljanih vozlišč za požarnim zidom ali ne. Eden od načinov omejevanja dostopa do omrežja je omejitev vseh povezav med procesi HPOM na upravljavskem strežniku in upravljanih vozliščih na ena omrežna vrata z uporabo komponente Communication Broker (slika 3.2) [5].



Slika 3.2: Komunikacija med HPOM in OA.

Za varno komunikacijo s strežnikom ali ostalimi agenti, mora agent na vozlišču imeti certifikat X509. Komunikacija se prične šele po izmenjavi certifikata, ki je podpisan z 1024-bitnim ključem. Glavne komponente, ki skrbijo za kreiranje in upravljanje s certifikati so:

- certifikatni strežnik, ki se nahaja na strežniku HPOM,
- OA Key Store,
- OA Certificate Client.

3.3.3 Varnost znotraj HP Operations Manager

HPOM skrbi za varnost na več različnih načinov [4]:

- Dostop do HPOM:
  Dostop do nadzorne plošče HPOM je dovoljen samo uporabnikom, ki so registrirani v HPOM.

- Varnost podatkovne baze:
  Podatkovna baza je dostopna tistim uporabnikom, ki imajo dostop za prijavo v operacijski sistem. Ko se uporabnik prijavi, varnostni mehanizmi podatkovne baze prevzamejo nadzor dostopa do podatkovne baze in njenih tabel. Te so določeni s konfiguracija baze in možnosti, ki jih ponuja uporabljena programska oprema podatkovne baze (Oracle, PostgreSQL).

- Zaganjanje aplikacij:
  Aplikacije se zaženejo z uporabniškim računom, ki smo ga navedli ob kreiranju njene konfiguracije. Ob zagonu komponenta Action Agent spremeni trenutnega uporabnika v navedenega in z njim zažene aplikacijo. V kolikor uporabnik nima ustreznih pravic, se aplikacija ne zažene.

- Oddaljene akcije:
  Oddaljene akcije so avtomatske ali operatersko akcije, ki se poženejo na upravljanem vozlišču. Ker se lahko pripeti, da bi se oddaljena akcija zagnala na vozlišču, kjer to ne želimo, nam HPOM ponuja različne varnostne mehanizme, s katerimi to preprečimo. Na vozliščih, kjer ne želimo izvajati oddaljenih akcij, jih lahko popolnoma preprečimo ali pa omejimo le na izvajanje tistih, ki prihajajo iz točno določenih IP naslovov.

3.4 Sestavni deli HP Operations Managerja

HPOM okolje sestavljajo upravljavski strežnik in upravljana vozlišča ter njihove komponente, kar je prikazano na sliki 3.3.

3.4.1 Upravljavski strežnik

Upravljavski strežnik HPOM ima osrednjo procesno vlogo. Vsebuje bazo, ki služi hranjenju podatkov o vseh sporočilih HPOM, nastavitvah HPOM, ter vseh potrebnih programskih paketih. Z uporabo teh podatkov, lahko tvorimo poročila za sistemske administratorje. Z uporabo poročil lažje definiramo navodila, ki pomagajo operatorjem pri reševanju problemov, ki jih sprožijo podobni dogodki in pri lažji avtomatizaciji procesov reševanja problemov [3]. Glavne naloge upravljavskega strežnika:

- zbiranje podatkov iz upravljanih vozlišč,
- upravljanje in združevanje sporočil v skupine,
- upravljanje z akcijami,
  - zagon lokalnih in oddaljenih akcij na ustreznem vozlišču,
  - proženje sej na vozlišču, na primer zagon virtualne konzole,
- upravljanje zgodovine prejetih sporočil in zagnanih akcij,
- posredovanje sporočil drugim upravljavski strežnikom,
- namestitev programske opreme,
  - namestitev OA na vozlišča,
  - namestitev in posodobitev konfiguracije na vozliščih.

3.4.2 Upravljano vozlišče

Upravljana vozlišča so računalniški sistemi, ki so nadzorovani s strani HPOM. HPOM upravlja vozlišča s komunikacijo z OA ter zaganjanjem njegovih procesov [3].
Glavne naloge upravljanih vozlišč so:

- Prestrezanje sporočil:
  Po namestitvi in zagonu OA bere dnevniške datoteke nadzira pasti SNMP in procese ter storitve ali pa prestreza sporočila, ki jih pošiljajo druge aplikacije, ki tečejo lokalno na vozlišču.

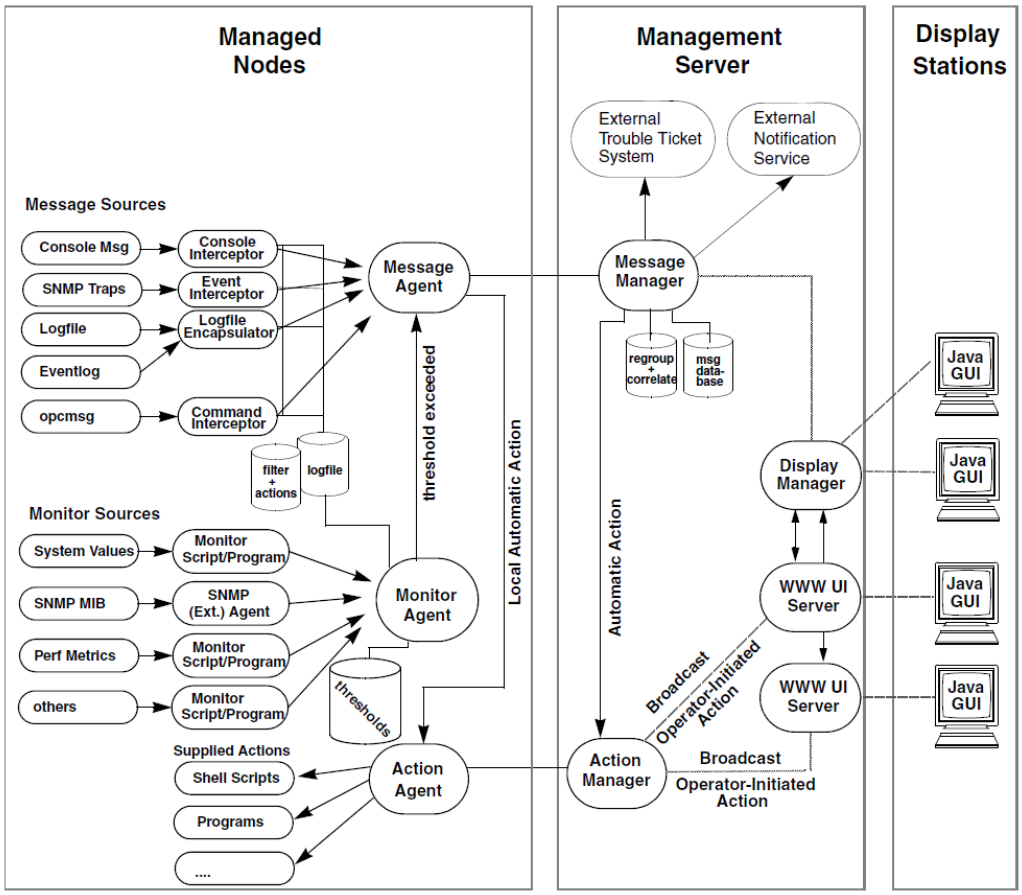
- Nadzorovanje zmogljivosti:
  Metrike zmogljivosti so nadzorovane na nastavljive intervale in tvorimo sporočila v primeru, ko se razlikujejo od nastavljenih vrednosti.

- Primerjanje sporočil:
  OA primerja vsa sporočila s pogoji v definiranih pravilih, ter posreduje nepredvidljiva ali pomembna sporočila upravljavskemu strežniku. Nastavimo lahko tudi izločanje podvojenih ali izključevanje med seboj podobnih dogodkov.

- Beleženje sporočil:
  Vsa sporočila so lahko beležena lokalno na vozlišču ali zapisana v podatkovno bazo na upravljavskem strežniku. To omogoča pregledovaje vseh sporočil, tudi tistih, ki smo jih zavrgli, ker so bila v tistem trenutku za operaterja nepomembna.

- Medpomnenje sporočil:
  V primeru, ko je upravljavski strežnik nedosegljiv, se sporočila zadržijo v medpomnilniku, dokler se strežnik ne odzove in lahko ponovno dostopa do njih.

- Odpravljanje problemov:
  Popravljalni ukrepi so lahko zagnani na vozliščih, kot odgovor na sporočilo.



Slika 3.3: Arhitektura okolja HPOM.

3.4.3 Dogodki

Dogodek je napaka oziroma incident, ki se zgodi na objektu v računalniškem okolju. Tipično predstavlja spremembo stanja ali prekoračitev praga nadzorovane vrednosti. Večina dogodkov predstavlja problem v sistemu, ki se je zgodil in za vsakega od njih lahko pošljemo sporočilo, ki pove kako napako odpraviti. Ker vsak dogodek predstavlja sporočilo, lahko hitro pride do preobremenitve upravljavskega strežnika ter zasičenja operaterja. Z uporabo korelacije, lahko ugotovimo povezave med sorodnimi dogodki in jih grupiramo v dogodkovne tokove, ki predstavljajo manjše lažje obvladljive informa-

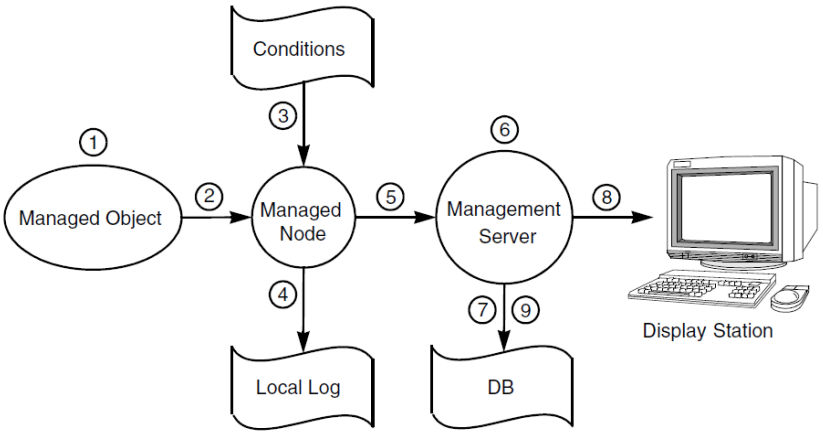
cije. Izločimo lahko dogodke, ki so sorodni in nadomestimo več posameznih sporočil z enim samim. Vse to se dogaja v realnem času.

3.4.4 Sporočila

Sporočila so strukturirani skupki informacij, ki jih sprožijo dogodki. Informacije, ki jih vsebujejo so po navadi izvorna aplikacija, ciljni objekt, resnost problema in tekstovni opis. Z uporabo ustreznih spremenljivk, pa lahko dodamo v sporočilo tudi dodatne poljubne informacije. HPOM lahko prestreza sporočila iz različnih izvorov:

- dnevniških datotek,
- nadzorovanih objektov,
- pasti SNMP,
- sporočil HPOM,
- uporabniških aplikacij.

Sporočila se lahko združujejo v logično povezane skupine. Skupina vsebuje sporočila s sorodnih virov in nudi informacije o določenih upravljanih objektih ali storitvah. Skupine lahko dodelimo tudi točno določenemu uporabniku in sporočila znotraj skupine bodo vidna le njemu. Slika 3.4 prikazuje potek pošiljanja in obdelovanja sporočila.



Slika 3.4: Potek pošiljanja sporočila.

1. Na nadzorovanem objektu se zgodi dogodek in kot posledica se ustvari sporočilo.

2. OA prejme sporočilo.

3. Na podlagi pogojev, ki jih določajo pravila, se lahko sporočila in njegove dvojnike zatre, ostala pa posreduje naprej.

4. Na upravljanem vozlišču lahko vsa sporočila beležimo v podatkovno bazo.

5. Sporočila so pretvorjena v format, ki ga HPOM razume ter posredovana upravljavskemu strežniku. Na podlagi njihovega tipa, se lahko zaženejo samodejne akcije.

6. Upravljavski strežnik obdela sporočila na enega izmed načinov:

   (a) samodejno dodeli sporočila drugi skupini sporočil,

   (b) samodejno zažene oddaljene akcije na vozlišču,

   (c) posreduje sporočilo zunanjim sistemom za poročanje,

(d) medpomni sporočilo.

7. Aktivna sporočila se shranijo v podatkovno bazo.

8. Sporočila se prikažejo v sporočilnem brskalniku v administratorski konzoli.

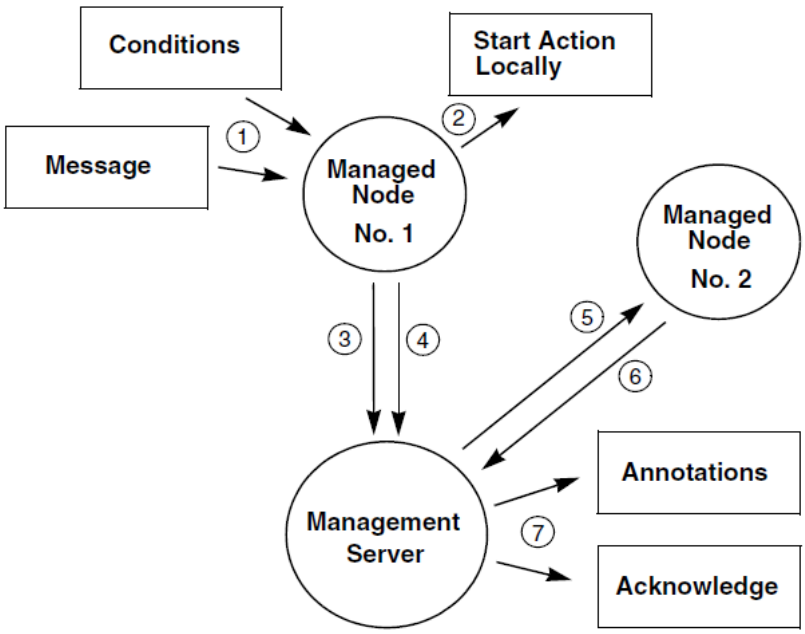
9. Po potrditvi, se sporočilo odstrani iz brskalnika in shrani v bazo zgodovine sporočil.

3.4.5 Akcije

Akcija je odgovor na sporočilo. V kolikor smo opredelili dogodek, kot problem, HPOM zažene popravljalno akcijo. Akcije lahko uporabljamo tudi pri vsakodnevnih opravilih, kot je na primer vsakodnevni zagon varnostnega kopiranja na vozlišču. Akcija je lahko skript, program, ukaz, zagon aplikacije oziroma poljuben odgovor. HPOM omogoča zagon samodejnih akcij, akcij, ki jih zažene operater, ter aplikacij.

3.4.5.1 Samodejne akcije

Samodejne akcije so definirani odgovori na sporočila. Ne potrebujejo interakcije operaterja in se zaženejo takoj, ko sporočilo prispe. Operator jih lahko ustavi ali pa ponovno zažene po potrebi (slika 3.5).



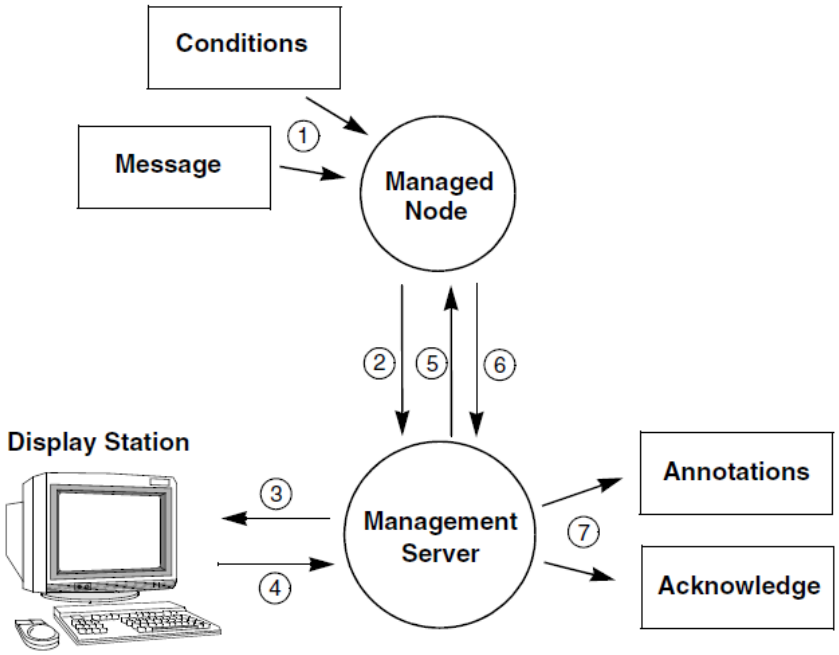
Slika 3.5: Potek delovanja samodejne akcije.

1. Sporočilo je prestreženo na vozlišču v skladu z definiranimi pogoji.
2. Če je cilj vozlišče A se akcija zažene lokalno.
3. Vozlišče A poroča o rezultatu akcije upravljavskemu strežniku.
4. Če je cilj vozlišče B, A obvesti upravljavski strežnik.
5. Upravljavski strežnik odgovori z navodili za zagon akcije.
6. Vozlišče B poroča o rezultatu upravljavskemu strežniku.
7. Po uspešno izvedeni akciji, se rezultate zabeleži in sporočilo samodejno potrdi.

3.4.5.2 Operaterske akcije

Kot samodejne akcije so tudi operaterske akcije definirane in povezane s sporočili. Razlikujejo se v tem, da jih operater zažene ročno. Uporabljajo

se kadar je prej potrebno izpolniti določene predpogoje. Na sliki 3.6 vidimo potek delovanja operaterske akcije.



Slika 3.6: Potek delovanja operaterskih akcij.

1. Sporočilo je prestreženo na vozlišču v skladu z definiranimi pogoji.
2. Sporočilo se posreduje upravljavskemu strežniku.
3. Sporočilo se pošlje ustreznemu operatorju v konzolo HPOM.
4. Operater zažene akcijo.
5. Navodila za zagon akcije se pošljejo vozlišču.
6. Vozlišče poroča o rezultatu akcije upravljavskemu strežniku.
7. Po uspešno izvedeni akciji, se rezultate zabeleži in sporočilo samodejno potrdi.

3.4.5.3 Aplikacije

Aplikacije so skripti ali programi, ki so integrirani v HPOM. Z razliko od samodejnih in operatersko akcij, niso direktno vezani na sporočila. Do njih ne moremo dostopati preko brskalnika sporočil, temveč so aplikacije orodja, do katerih dostopamo preko menija v administracijski konzoli.

3.4.6 Uporabniki

HPOM razlikuje med dvema uporabniškima vlogama:

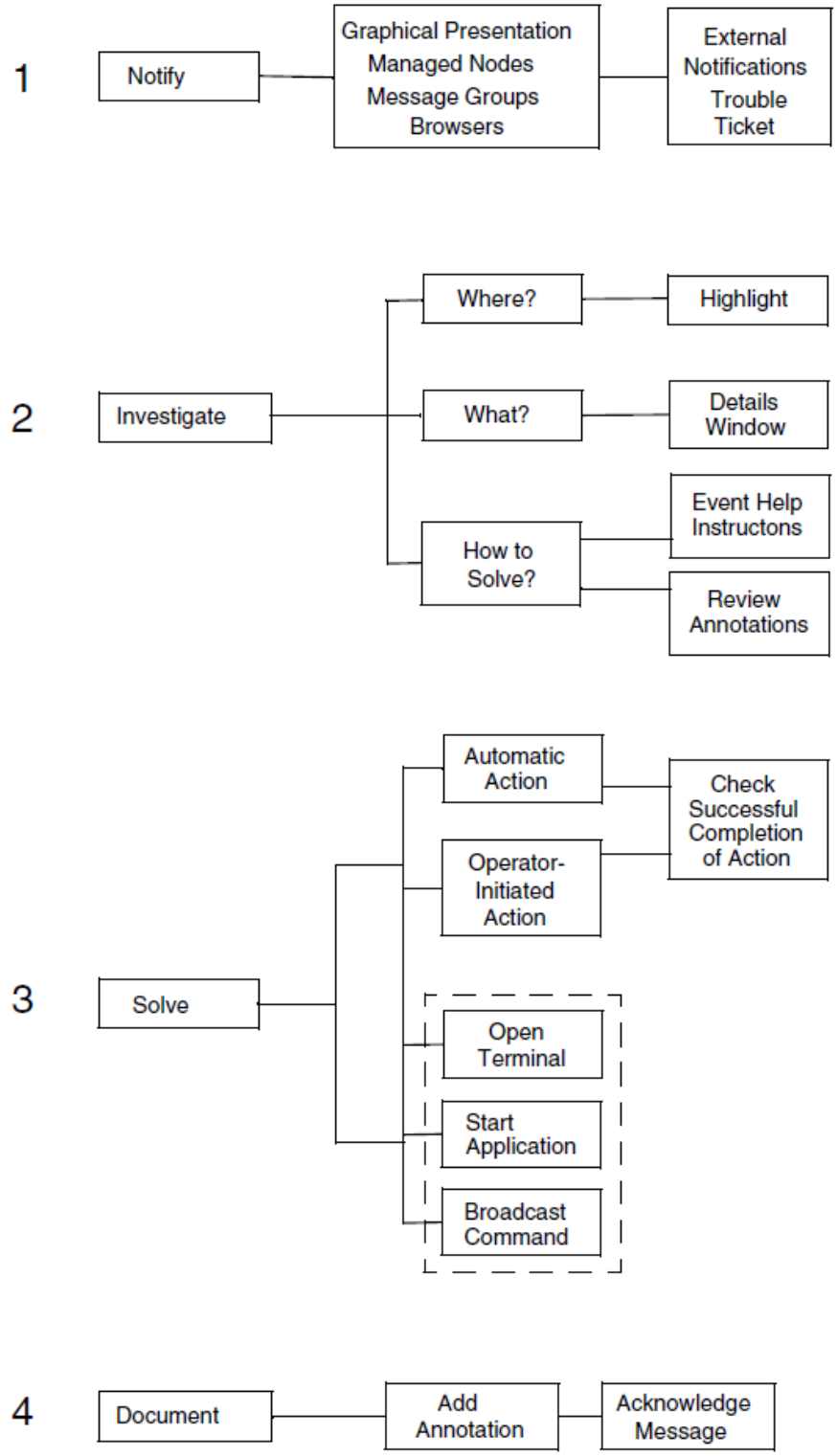
- Administrator HPOM je uporabnik, ki ima neomejene pravice. Uporablja se pri namestitvi, vzdrževanju in konfiguraciji programske opreme HPOM, skrbi pa tudi za prilagoditev uporabniških okolij in delegacijo nalog operaterjem.

- Operater HPOM je uporabnik, ki vzdržuje, upravlja in nadzira sistem ter njegove nadzorovane objekte. HPOM dovoljuje več operaterjev na istem sistemu. Vsak ima lahko določene odgovornosti in zmožnosti glede na njegovo poznavanje domene. Ustvarimo lahko abstraktne uporabniške profile, ki imajo določen nabor pravic in jih kasneje dodelimo fizičnim uporabnikom.

V kolikor pa imamo opravka z majhnimi okolji je lahko administrator in operater kar ista oseba [4].

3.5 Postopek odpravljanja napak s HP Operations Manager

Diagram poteka odpravljanja napak z uporabo HPOM [3] prikazuje slika 3.7.



Slika 3.7: Postopek reševanja napak s HPOM.

1. Obveščanje:
   OA nadzoruje aktivnosti na sistemu. Če pride do napake, obvesti uporabnika na enega od naslednjih načinov:

   - pošlje sporočilo upravljavskemu strežniku,
   - obarva ikono vozlišča, skladno z resnostjo problema,
   - obarva polje resnosti problema v sporočilnem brskalniku, oziroma celotno vrstico sporočila, če je tako nastavljeno,
   - prikaže sporočilo in njegove atribute (na primer čas pošiljanja, status akcije itd.),
   - posreduje sporočilo zunanjemu sistemu za sledenje težavam.

2. Preiskovanje:
   Razumenvanje problema ter njegovega vzroka. V večjih okoljih je ključnega pomena hitro izslediti problem, ter ga rešiti.

3. Reševanje:
   HPOM nam ponuja različne načine izvajanja popravljalnih ukrepov pri reševanju problema.

   - Samodejne akcije:
     HPOM zažene samodejno akcijo takoj ko prejme sporočilo o napaki. Samodejne akcije lahko uporabnik požene tudi ročno takrat, ko je to potrebno.
   - Operaterske akcije:
     Operater ročno zažene popravljalno akcijo, ko prejeme sporočilo o napaki.
   - Navodila uporabniku:
     Sporočila o napaki, lahko opremimo s specifičnimi napotki, kako problem odpraviti.
   - Administracijska konzola:
     Uporabimo lahko administracijsko konzolo za zaganjanje različnih

aplikacij, razpršimo ukaze po več sistemih ali odpremo virtualno konzolo neposredno na prizadetem sistemu, ter tako izvajamo popravljalne ukrepe.

4. Dokumentiranje:
   Rešimo problem, ter dokumentiramo rešitev na katero se lahko sklicujemo v prihodnosti.

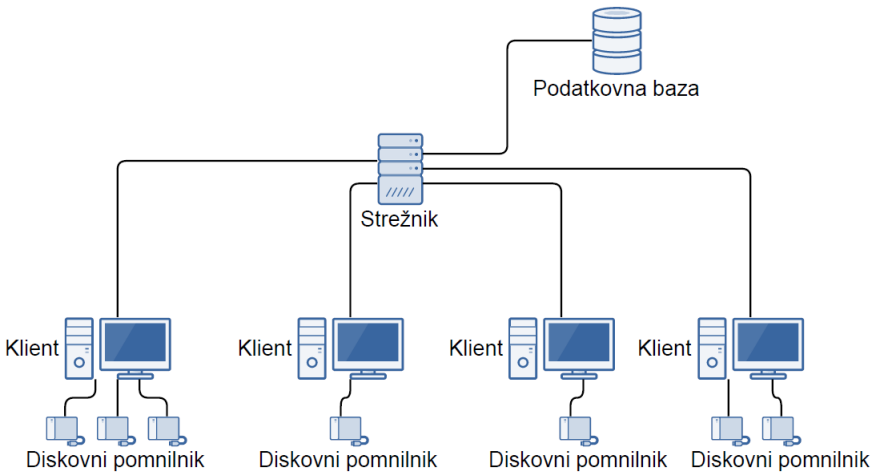
Poglavje 4

Opis aplikacije za upravljanje diskovnih pomnilnikov

Razvoja vtičnika SPI si bomo ogledali na primeru aplikacije GDM v velikih distribuiranih poslovnih okolij. Glavne značilnosti naše aplikacije so:

- razširljiva in prilagodljiva arhitektura,
- podpora različnih računalniških okolij (Windows, Linux, UNIX),
- centralna administracija,
- zaščita podatkov in komunikacije med napravami,
- nadzorovanje, poročanje in obveščanje o pravilnosti delovanja trdih diskov,
- pregledovanje trdih diskov,
- nadzor zmogljivosti,
- nadzor S.M.A.R.T. parametrov,
- defragmentacija,
- brisanje, formatiranje in kreiranje particij.

Podatkovno hranjenje je v računalništvu zelo obširno področje. Obsega lahko različne tipe pomnilnikov; trde diske, magnetne trakove, optične medije in različne vrste implementacij; omrežno hranjenje, diskovna polja, samostojni diski itd [8]. Za prikaz delovanja in razvoja vtičnika SPI, ne bomo potrebovali podrobnega znanja s področja delovanja in implementacije diskovnih shramb in trdih diskov, zato se bomo obnašali kot, da aplikacija deluje kot črna škatla v katero nimamo vpogleda in njeno funkcionalnost poenostavili. Uporabljali bomo izraz disk, ne bomo pa podrobno ločevali med tipi diskov (HDD, SSD, VHD). Osredotočili se bomo na klasične namizne trde diske in zanemarili omrežne shrambe, diskovna polja in virtualne diske.
Arhitektura naše rešitve GDM (slika 4.1) je sestavljena iz enega ali več strežnikov, na katerega so povezani odjemalci, ki predstavljajo različne računalniške sisteme, ter naprave, ki predstavljajo diske, ki jih odjemalci vsebujejo. Na strežniku tečejo podporni procesi ter storitve, ki skrbijo za delovanje aplikacije in obdelavo podatkov in podatkovna baza, ki hrani meta podatke. Na odjemalcih je nameščena ustrezna programska oprema, ki skrbi za varno povezavo in komunikacijo s strežnikom. Tabela 4.1 prikazuje komponente GDM infrastrukture in njihove vloge.



Slika 4.1: Arhitektura GDM aplikacije.

Komponenta	**Vloga**
Strežnik	Osrednji sistem, kjer je nameščena ključna programska oprema GDM, s katerega se vršijo vse aktivnosti upravljanja in nadziranja diskov.
Podatkovna baza	Podatkovna baza, ki hrani podatke o informacijah naprav in odjemalcev, kakšen je status njihovih parametrov in podatke o konfiguraciji.
Odjemalec	Sistem, ki vsebuje naprave.
Naprave	Naprave, ki predstavljajo diskovni pomnilnik.
Storitve	Storitve za delovanje in komunikacijo aplikacije z odjemalci in podatkovno bazo.

Tabela 4.1: Komponente GDM okolja.

Poglavje 5

Vtičnik Smart Plugin

SPI je vtičnik, ki razširja HPOM z dodatno funkcionalnostjo. Omogoča nam samodejno nadzorovanje infrastrukture ali aplikacij, ki jih HPOM privzeto ne zna. Vtičnik SPI nudi uporabniku retroaktivno in proaktivno nadzorovanje, zaznavanje in reševanje problemov v specifični IT domeni. Povečuje produktivnost uporabnika z optimizacijo in avtomatizacijo različnih nalog in tako zmanjšuje čas reševanja problemov. SPI omogoča tudi nadzorovanje dosegljivosti in zmogljivosti delovanja v domeni delovanja IT.

Vtičnik SPI je tipično sestavljen iz naslednjih komponent:

- Servisne hierarhije, ki predstavlja grafični prikaz - topologijo nadzorovane infrastrukture.
- Pravil, ki jih namestimo na upravljano vozlišče in skrbijo za prestrezanje in posredovanje sporočil o napakah na strežnik HPOM.
- Instrumentacije, ki vsebuje skripte ali datoteke, ki jih uporablja OA za odkrivanje in nadzorovanje virov na katere se sklicujemo v pravilih. Instrumentacija zbira podatke in odloča ali je potrebno poslati opozorilo. Z izrazom instrumentacija bomo označili tudi našega agenta za zbiranje podatkov in vse podporne skripte, ki jih bo naš vtičnik SPI uporabljal.

- Aplikacij, s katerimi nudimo uporabniku poganjanje podpornih programov, ki mu nudijo pomoč pri nadzorovanju ali izvrševanju popravljalnih akcij.

- Ostale konfiguracije, ki povezuje vtičnik SPI v zaključeno celoto. V to kategorijo sodijo konfiguracijske datoteke za registracijo procesa v kontrolno komponento HPOM, združevanje definicij sporočil, pravil in vozlišč v skupine, ter dodeljevanje pravic uporabnikom.

5.1 Razvoj vtičnika Smart Plugin

Preden se lotimo razvoja vtičnika SPI moramo zelo dobro poznati aplikacijsko domeno, ki jo bomo nadzorovali. Tako lahko razvijemo SPI, ki bo omogočal kvaliteten nadzor in predvidel težave še preden se zgodijo. Poznati moramo komponente aplikacije, storitve ki jih ponuja in relacije med njimi. Seznanjeni moramo biti tudi s tipičnimi scenariji namestitve, ki jih priporoča razvijalec in jih stranke uporabljajo. Ko domeno poznamo, naredimo model s katerim identificiramo objekte, ki jih bomo nadzorovali. Nato definiramo parametre zdravja, ki jih bomo spremljali. Vtičnik SPI, ki ga razvijamo mora uporabniku omogočati efektivno reševanje problemov. Dogodki, ki jih bomo tvorili, morajo nuditi izvajanje popravljalnih akcij in uporabniku jasno sporočati, zakaj je do njih prišlo. To naredimo tako, da dogodke opremimo z navodili, ki opisujejo kakšen je problem, kaj naj uporabnik stori in kje lahko najde več informacij o nastali situaciji. Za dogodke bomo definirali samodejne akcije, ki bodo skrbele za reševanje enostavnih problemov. Na primer, če se določena storitev ustavi, se lahko samodejno ponovno zažene in uporabnika opozorimo le, kadar tudi ponovni zagon ne uspe. Zaradi lažje preglednosti, moramo poskrbeti, da za isti dogodek ne pošiljamo več istih sporočil, saj so v večini primerov nepotrebna. Prav tako moramo vsak dogodek označiti s pravilno stopnjo resnosti, ki jo težava predstavlja.

Priporočila za stopnje resnosti so sledeča [4]:

- Kritična - Prekiniti trenutno aktivnost in odpraviti problem.
- Zelo resna - Problem je potrebno rešiti v naslednji 6-12 urah. V nasprotnem primeru lahko povzroči izpad delovanja.
- Resna - Problem je potrebno rešiti v naslednjih 12-24 urah.
- Opozorilna - Problem je potrebno rešiti v enem tednu.
- Informativna - Problem je potrebno potrditi.

Razvoj vtičnika SPI razdelimo na tri dele. V prvem delu razvijemo agenta, ki skrbi za zbiranje podatkov na upravljanih vozliščih. V drugem koraku implementiramo konfiguracijo HPOM, ki ustrezno interpretira zbrane podatke. Konfiguracija HPOM definira pravila, ki skrbijo za prestrezanje sporočil, stanj zdravja objektov, akcije, sporočilne skupine in grafični prikaz topologije. V zadnjem koraku našega agenta registriramo kot komponento, ki je upravljana s strani HPOM.

5.2 Zahteve delovanja vtičnika Smart Plugin

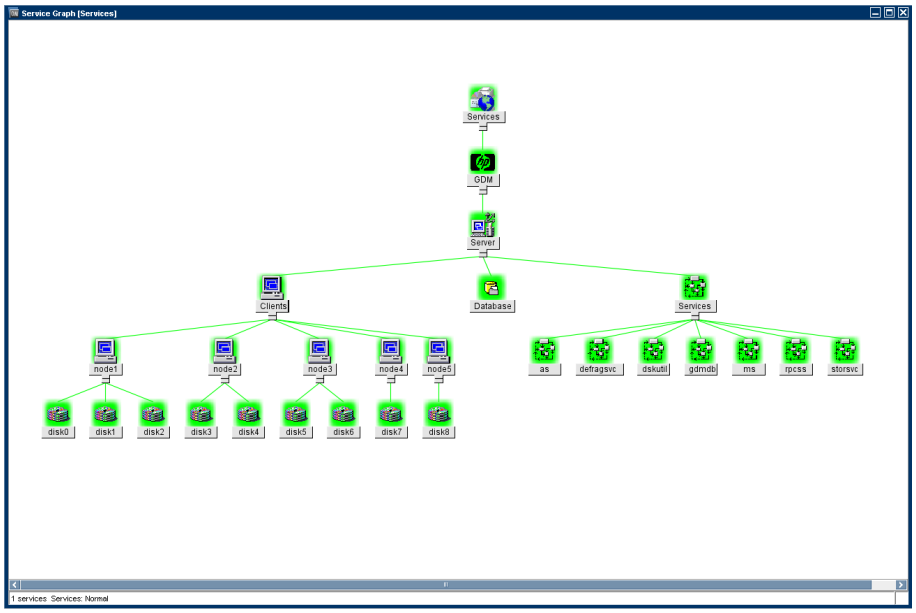
Z vtičnikom SPI zbiramo podatke za grafični prikaz, nadzorovanje infrastrukture in delovanje diskovnih pomnilnikov, ter ob zaznanih nepravilnostih te ustrezno poročamo uporabniku preko sporočil in stanj zdravja objektov. Zahteve razdelimo na štiri večja področja funkcionalnosti in za vsako področje opišemo zahteve delovanja vtičnika SPI.

5.2.1 Grafični prikaz topologije okolja GDM

V administratorski konzoli HPOM smo v pogledu Service Navigator prikazali topologijo (slika 5.1). Prikazali smo vse objekte, ki sestavljajo GDM okolje ter relacije med njimi, v drevesni strukturi. Odkrivanje topologije se izvaja na vsakem upravljanem vozlišču, kjer je nameščena kakšna od komponent

aplikacije GDM. Vozlišče je lahko strežnik, odjemalec ali baza. Ker se okolje spreminja - dodajamo nove naprave, odjemalce ali strežnike - se morajo spremembe ustrezno odražati tudi v grafičnem prikazu. Posodabljanje se vrši samodejno, ko sistem zazna spremembo, ročno preko ustrezne akcije ali pa periodično preverjamo na vnaprej določen interval, če je do spremembe prišlo. Drevesna struktura prikaza topologije je sestavljena iz naslednjih elementov:

- korenski element,
- strežnik,
- podatkovna baza,
- skupina odjemalcev,
- odjemalec,
- naprava,
- skupina storitev,
- storitve.



Slika 5.1: Topologija okolja GDM.

5.2.2 Nadzorovanje objektov in prikaz zdravja v topologiji

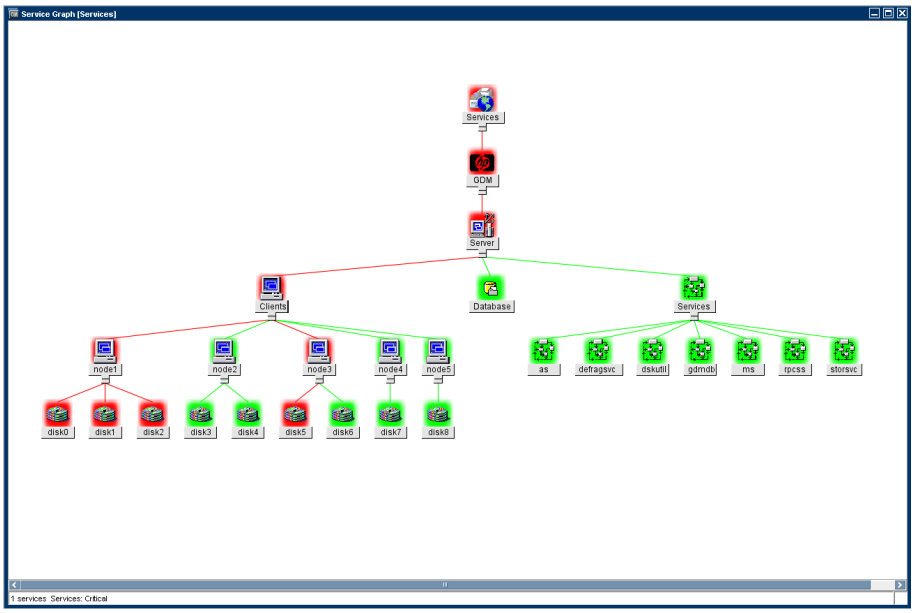
Za objekte želimo nadzorovati:

- stanje podatkovne baze,
- stanje odjemalcev,
- stanje storitev,
- stanje diskov,
- S.M.A.R.T. parametre diskov,
- zasedenost prostora na diskih,
- število slabih sektorjev,
- zmogljivosti diskov.

Za vsak prikazan objekt v topologiji nadzorujemo stanje in ostale ustrezne parametre. Če zaznamo odstopanja od normalnih vrednosti ali nedelovanje katerega izmed njih to prikažemo kot spremembo stanja zdravja in jo označimo z ustrezno barvo.
Ločimo med stanji:

- informativno - zelena,
- opozorilno - modra,
- resno - rumena,
- zelo resno - oranžna,
- kritično - rdeča.

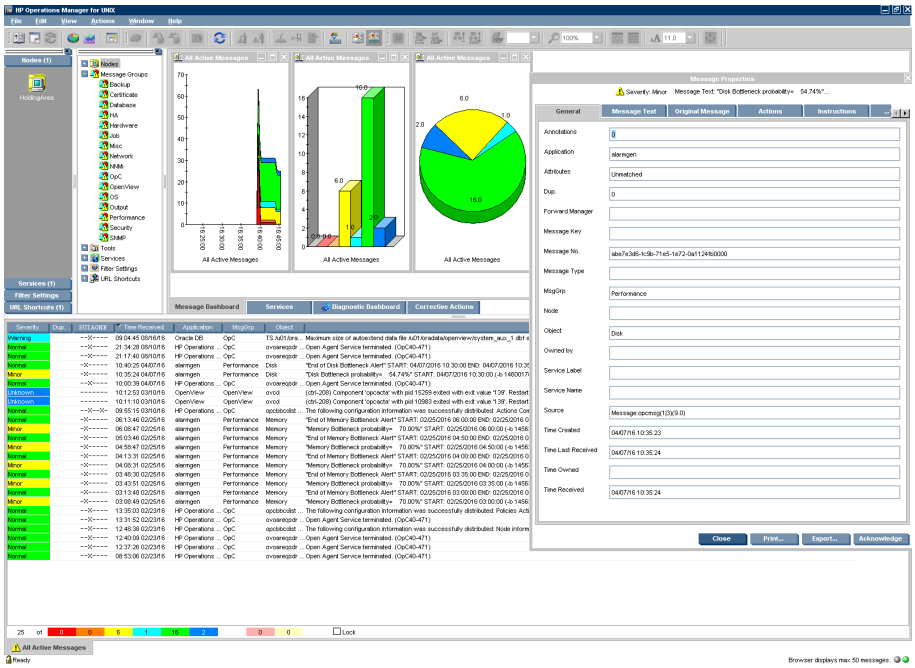
V kolikor v topologiji objekt spremeni zdravje, stanje propagiramo po drevesu do korenskega elementa in vse njegove starše obarvamo v barvo stanja objekta, kjer je do spremembe prišlo, kar prikazuje slika 5.2. Na ta način dosežemo, da uporabniku ni potrebno spremljati celotnega drevesa ampak le par zgornjih nivojev. S tem povečamo preglednost nad zelo velikimi okolji, ki morda zaradi razvejanosti ne bi bili prikazani v celoti.



Slika 5.2: Zdravje objektov v GDM topologiji.

5.2.3 Pošiljanje sporočil in dogodkov ob zaznanih nepravilnostih

Za vse nadzorovane objekte moramo ob primeru nepravilnosti poslati ustrezno sporočilo, ki opisuje do kakšnega problema je prišlo ter na katerem objektu. Sporočilo vsebuje stopnjo resnosti problema ter ustrezne napotke, kako problem rešiti. Sporočila zaradi boljše preglednosti združujemo v skupine. Imamo skupino sporočil za odjemalce, naprave, storitve, dogodke in interna sporočila. Sporočila si lahko uporabnik ogleda v sporočilnem brskalniku v administratorski konzoli, ki je prikazan na sliki 5.3.



Slika 5.3: Sporočilni brskalnik administracijske konzole.

5.2.4 Proženje popravljalnih in drugih akcij v GDM okolju

Vtičnik SPI mora omogočati zaganjanje popravljalnih in podpornih akcij, s katerimi lahko ustrezno reagiramo na morebitne napake v okolju ali pa pridobimo podrobne podatke in poročila o delovanju sistema. Definirali smo naslednje akcije in orodja:

- posodobitev grafičnega prikaza topologije,
- sprememba konfiguracijskih parametrov vtičnika SPI,
- poročilo informacij o odjemalcu,
- poročilo o podatkovni bazi,
- prikaz neuporabljenih naprav,

- poročilo napak naprave,
- poročilo o stanju naprave,
- zagon defragmentacije,
- kreiranje particij,
- pregledovanje naprave,
- testiranje naprave,
- brisanje in formatiranje naprave,
- stanje storitev,
- zagon storitve,
- zaustavitev storitve.

5.3 Uporabljene tehnologije

Agent, ki skrbi za pridobivanje podatkov o okolju GDM, je implementiran v programskem jeziku Perl. Za Perl smo se odločili, saj je vključen v instalacijske pakete HPOM in OA in tako na voljo ob namestitvi, tako na strežniku kot upravljanem vozlišču. S tem izključimo možnost neskladnosti verzij in morebitnih težav zaradi manjkajočih Perl modulov (programskih knjižnic), ki jih uporabljamo pri implementaciji agenta. Lahko se zanesemo na to, da bo naš agent deloval na vseh okoljih, kjer je nameščen HPOM ali OA. S Perl programskim jezikom zadostimo potrebam po delovanju vtičnika SPI na različnih operacijskih sistemih, saj nudi podporo različnim okoljem. Aplikacija GDM hrani informacije o delovanju diskov, zmogljivosti, S.M.A.R.T. parametrih, seznamu odjemalcev, naprav ter strežnikov, stanju podatkovne baze in podatke o ostalih podpornih procesih. Do njih dostopamo preko branja ustreznih datotek in različnih ukazov vmesnika ukazne vrstice, ki jih

nudi. Podatke zberemo preko Perl skripta, jih ustrezno prečistimo, obdelamo, strukturiramo in zapišemo v datoteko XML. Do te enostavno dostopamo preko Perl modula za delo z datotekami XML in jo uporabljamo za nadaljnje delo. Na vsakem vozlišču, kjer teče agent bomo preko vmesnika ukazne vrstice OA pošiljamo sporočila na strežnik HPOM, ki uporabnika obveščajo o stanju ter napakah v okolju.

5.3.1 Perl

Perl je višje nivojski interpretiran skriptni jezik, ki ga ja razvil Larry Wall konec 80. let. Na razvoj Perl sta v veliki meri vplivala programski jezik C, ter Shell skriptni jezik. Perl se danes najpogosteje uporablja kot skriptni jezik za spletne strežnike ter sistemsko in omrežno administracijo. Zaradi njegove prilagodljivosti in enostavne uporabe ga najdemo tudi na drugih različnih področjih, kot so bančništvo, bioinformatika, ter grafično programiranje. Perl značilnosti olajšujejo programerjevo delo, vendar pogosto na račun večjih procesorskih ali pomnilniških zahtev. Vsebuje avtomatsko upravljanje s pomnilnikom, dinamične tipe, kompleksne podatkovne strukture, regularne izraze in itd. Uporabniku ne vsiljuje programerskih paradigem - objektnega, proceduralnega ali funkcijskega programiranja in tudi ne zahteva, da izbira med njimi. Perl odlikuje hitro in enostavno delo z besedilom, kakršnih je v programiranju večina problemov. Perl je jezik, s katerim lahko opravimo vsakršno nalogo, hitro in enostavno [10].

5.3.2 XML

XML je preprost, računalniški označevalni jezik, ki opisuje podatke v obliki, ki je človeško in strojno berljiva. XML podatki so samo opisljivi in samo definirani, saj je struktura vključena v opis podatkov, z uporabo značk. XML oblika se uporablja povsod, kjer želimo doseči konsistentno strukturirano predstavitev podatkov ali pa jih deliti z drugimi uporabniki [16].

5.3.3 S.M.A.R.T.

S.M.A.R.T. je tehnologija, ki jo vsebujejo diskovni pomnilniki in omogoča zaznavanje in poročanje številnih indikatorjev o zanesljivosti delovanja. Nadziranje poteka preko različnih metod in tipal, ki so vgrajeni v disk in njegov krmilnik. Namen S.M.A.R.T. tehnologije je napovedovanje strojnih napak preden se zgodijo in tako preprečuje izgubo podatkov. Napake pri trdih diskih ločimo v dve kategoriji: predvidljive in nepredvidljive. Predvidljive napake so rezultat postopne mehanične obrabe trdega diska, magnetne plošče, bralno pisalne glave ali katere izmed drugih komponent. Nadzorovanje lahko predvidi, kdaj se bodo te napake zgodile in nas pravočasno o tem obvesti. Nepredvidljive napake pa se zgodijo brez predhodnih opozoril in so lahko odraz okvare elektronskih komponent ali nenadne mehanične napake, ki se zgodi zaradi nepravilnega ravnanja z diskom. Statistično približno 60 odstotkov diskov odpove zaradi mehaničnih napak, ki pridejo zaradi postopne obrabe [8]. Indikatorji, ki kažejo na odpoved, so po navadi povišana temperatura delovanja, povišana glasnost, težave pri branju ali zapisovanju podatkov in povečano število okvarjenih sektorjev. Najbolj okrnjena implementacija S.M.A.R.T., ki poroča o statusu atributov deluje dvojiško. Ali je bil prag presežen ali ne in sporoča ali je disk v okvari ali pa deluje v redu. V naprednih implementacijah pa lahko zbiramo tudi podatke o najvišjih oziroma najnižjih doseženih vrednostih [17]. Danes skoraj vsi diski vsebujejo S.M.A.R.T. tehnologijo in merijo številne različne atribute. Uporabnik lahko nadzira te atribute s pomočjo različnih orodij in programskih rešitev.

5.3.4 Vmesnik ukazne vrstice in datoteke aplikacije za upravljanje diskovnih pomnilnikov

Opisi ukazov s katerimi bomo pridobivali informacije o okolju iz aplikacije GDM:

- `gdmserverinfo`
  Prikaže informacije o konfiguraciji strežnika GDM.

- `gdmservice`
  Zažene, ustavi ali prikaže status storitev ali procesov GDM.

- `gdmdbcheck`
  Preveri trajnost in pravilnost podatkovne baze.

- `gdmreport`
  Generira razna poročila o okolju GDM. Na primer o zasedenosti, zmogljivosti, in S.M.A.R.T. atributih naprav.

- `gdmdb`
  Vrši poizvedbe po podatkovni bazi.

- `gdmdiskscan`
  Sproži skeniranje diska.

- `gdmclientcheck`
  Preveri DNS povezavo med strežniki in odjemalci in izpiše informacije o odjemalcih.

- `gdmdevicestatus`
  Izpiše status diska.

Opisi datotek iz katerih bomo pridobivali informacije o okolju GDM:

- `gdm_log.txt`
  Centralna dnevniška datoteka dogodkov v okolju GDM.

- `gdm_device_info.dat`
  Datoteka z zapisi o napravah.

- `gdm_client_info.dat`
  Datoteka z zapisi o odjemalcih.

- `gdm_server_info.dat`
  Datoteka z zapisi o strežnikih.

5.3.5 Vmesnik ukazne vrstice HP Operations Agent

Opisi uporabljenih ukazov za delo z OA [4, 11].

- `opcmsg`
  Ukaz za pošiljanje sporočil z OA na strežnik HPOM. Dogodek, ki ga pošljemo z uporabo `opcmsg` se najprej interpretira s strani HPOM komponente Message Interceptor na upravljanem vozlišču. Sporočila so nato posredovana na strežnik HPOM, kjer jih procesirajo pravila Message Interface Policies. Ukaz `opcmsg` [11] zgenerira sporočilo z naslednjimi podatki:
  - `severity` - Označuje stopnjo resnosti problema. Vrednosti, ki so na voljo so critical, major, minor, warning in normal.
  - `application` - Aplikacija, ki je tvorila dogodek. Uporablja se v pravilih za izbiro ustreznega sporočila. Ime aplikacije mora biti unikatno.
  - `object` - Objekt, ki je zaznal dogodek.
  - `msg_text` - Besedilno sporočilo, ki ga želimo prikazati uporabniku.
  - `msg_grp` - Skupina v katero želimo uvrstiti sporočilo.
  - `node` - Vozlišče na katerem je prišlo do dogodka.

  Poleg vnaprej določenih argumentov, pa lahko pošljemo tudi poljubne vrednosti z uporabo `option` argumenta.
  - `-option spremenljivka=vrednost`

  Primer uporabe ukaza `opcmsg`:
  ```
  opcmsg severity=critical msg_grp=OS application=diskwatcher.sh
  object=/usr/lib node=ovtest.bbn.hp.com msg_txt=''Only 5% of
  entire disk space still available!''
  ```

- `opcsvcattr`
  Ukaza za nastavljanje atributov objektov.

- `opcnode`
  Ukaz za upravljanje z vozlišči.

- `opcragt`
  Upravljanje s procesi, ki tečejo na vozliščih, na daljavo s strežnika HPOM.

- `ovdeploy`
  Ukaz za upravljanje z vsebino. Omogoča pošiljanje podatkov med strežnikom HPOM in OA in izvajanje ukazov na daljavo.

- `ovconfget`
  Nastavlja ali pridobiva parametre shranjene v konfiguracijskih datotekah HPOM.

- `ovcreg`
  Registrira komponento v OA.

- `ovc`
  Zažene, ustavi ali izpiše status komponent OA.

5.4 Razvoj agenta za pridobivanje podatkov iz aplikacije za upravljanje diskovnih pomnilnikov

V tem poglavju je tehnično predstavljen razvoj vtičnika SPI po zahtevah, ki smo jih definirali. Opisani so tudi vsi programski vmesniki, preko katerih pridobivamo in posredujemo podatke med GDM, agentom, OA in strežnikom HPOM.
Ob zagonu `gdm_agent.pl` skripta se najprej požene `INIT` blok kode, ki inicializira strukturo XML in kreira `gdm_agent_out.xml` datoteko v kolikor ta še ne obstaja. V njej hranimo vse podatke, ki jih potrebujemo za opis okolja GDM.

XML vsebuje korenski element, ki označuje ime naše aplikacije. V naslednjem nivoju opišemo strežnik z njegovim identifikatorjem. Strežnik vsebuje domensko ime ter dodatne atribute, podatke o stanju podatkovne baze, odjemalce ter njihove atribute, naprave in seznam storitev, ki tečejo na strežniku. Ker so nam storitve, ki jih nadzorujemo znane vnaprej, njihove podatke vnesemo že ob inicializaciji. Zadnja sekcija `gdm_agent_out.xml` pa predstavlja vrednosti konfiguracijksih atributov, ki opisujejo delovanje vtičnika SPI. Z njimi lahko nastavljamo intervale, kako pogoste se vršijo koraki odkrivanja objektov, preverjanja zdravja, nadzorovanja dogodkov, nadzorovanja objektov, posodabljanja topologije in nastavitve o beleženju v dnevniško datoteko. Vsebina gdm_spi_data.xml datoteke po inicializaciji je prikazana v dodatku A.1.

Po klicu `INIT` bloka se izvrši `main` funkcija našega programa. Sestavljena je iz klicev petih korakov, ki tečejo v neskončni zanki.

5.4.1 Korak odkrivanja objektov

V prvem koraku najprej odpremo in preberemo `gdm_spi_data.xml` datoteko, za kar uporabimo Perl modul XML::Simple s katerim enostavno operiramo nad strukturami XML. Glavni funkciji iz knjižnice XML::Simple, ki ju bomo uporabljali sta XMLIn, ki prebere datoteko XML v Perl podatkovno strukturo, ter XMLOut ki podatkovno strukturo zapiše v datoteko XML. Ko smo prebrali datoteko preverimo, če je nastopil čas za odkrivanje objektov, saj se koraki vršijo na časovne intervale. To preverimo tako, da iz `gdm_spi_data.xml` preberemo atribut `valid` v znački `discovery`. Atribut hrani čas od zadnjega koraka za odkrivanje objektov, kateri je prištet časovni interval v sekundah, ki se nahaja v atributu `interval`. Če je trenutni čas večji kot `valid` čas, s korakom nadaljujemo, drugače pa ga preskočimo. Sledi vpisovanje podatkov o strežniku GDM. Njegov identifikator zgradimo iz domenskega imena strežnika na katerem teče naš skript in ga pretvorimo v GUID identifikator s pomočjo MD5 zgoščevalne funkcije. V `gdm_agent_out.xml` nastavimo tudi podatke o domenskem imenu in operacijskem sistemu, ki jih dobimo z vgrajenimi Perl funkcijami. Preverjanje

nadzorovanih storitev, ki tečejo na strežniku naredimo tako, da izvršimo ukaz vmesnika ukazne vrstice `gdmservice -status`, ki nam vrne podatke o njih:

```
ProcName          Status   [PID]
=================================
as                : Active   [1957]
ms                : Active   [1955]
rpcss             : Active   [1956]
dskutil           : Active   [1952]
defragsvc         : Active   [1951]
gdmdb             : Active   [1987]
storsvc           : Active   [1988]
```

Ker so podatki v neustrezni obliki jih moramo najprej prečistiti. Za to uporabimo regularne izraze in uporabimo le imena storitev v stolpcu `ProcName`. V primeru, da katera izmed storitev v datoteki XML ni navedena v seznamu storitev, ki nam jih je vrnil ukaz `gdmservice -status` jo izbrišemo iz strukture XML in zabeležimo v dnevniški datoteki, da smo jo odstranili. Podoben postopek uporabimo tudi pri zapisih o odjemalcih in napravah. Odjemalce pridobimo iz datoteke `gdm_client_info.dat`, ki vsebuje njihove podatke:

```
-host "node1.gdm.com" -os "gpl x86_64 linux -2.6.32-504.el6.
    x86_64" -core 12.03 -as 12.03
-host "node2.gdm.com" -os "win" -core 12.03 -as 12.03
-host "node3.gdm.com" -os "win" -core 12.03 -as 12.03
-host "node4.gdm.com" -os "gpl x86_64 linux -2.6.32-504.el6.
    x86_64" -core 12.03 -as 12.03
-host "node5.gdm.com" -os "gpl x86_64 linux -2.6.32-504.el6.
    x86_64" -core 12.03 -as 12.03
```

Vsaka vrstica se nanaša na enega odjemalca, podatki pa so prikazani v obliki:

```
'''-ime_atributa' 'vrednost_atributa'''
```

Podatke o odjemalcih v `gdm_agent.pl` pretvorimo v obliko:

```
ime_atributa=''vrednost_atributa''
```

in jih zapišemo v `gdm_agent_out.xml`. Seznam naprav in njihovih podatkov pridobimo iz datoteke `gdm_device_info.dat`, ki vsebuje zapise v obliki:

GUID	"fe362500-e033-442c-bfc5-dc907120c8f3"
NAME	"Disk0"
DESCRIPTION	"Local Disk"
HOST	"node1.gdm.com"
FILESYSTEM	"NTFS"
TYPE	"HDD"
POOL	""
LIBRARY	""

Vrednosti ustrezno pretvorimo ter jih shranimo v ustrezne atribute v datoteki XML. Posodobimo podatke kdaj smo končali z odkrivanjem objektov. V atribut `time` vpišemo trenutni čas, v atribut `valid` pa trenutni čas, ki mu prištejemo nastavljeno vrednost za časovni interval. Vse podatke, ki smo jih zbrali vpišemo v datoteko XML in s tem končamo korak s katerim smo odkrili objekte, ki jih bomo v naslednjih korakih nadzorovali.

5.4.2 Korak preverjanje zdravja objektov

Preberemo datoteko XML in preverimo ali ustrezamo časovnemu okviru za izvedbo koraka za preverjanja zdravja objektov na isti način, kot smo to naredili v prejšnjem koraku. Najprej bomo dodali storitvam njihove statuse. Ponovno uporabimo ukaz `gdm_service -status` vendar sedaj uporabimo stolpec `Status [PID]`. PID številko v oglatih oklepajih odstranimo in v strukturo storitvam dodamo v atribut `status` in ustrezno vrednost stanja.

```
ProcName              Status   [PID]
=============================================
as                  : Active   [1957]
ms                  : Active   [1955]
rpcss               : Active   [1956]
dskutil             : Active   [1952]
defragsvc           : Active   [1951]
gdmdb               : Active   [1987]
storsvc             : Active   [1988]
```

Preverjanje zdravja podatkovne baze ugotovimo s pomočjo ukaza `gdmdbcheck`, ki nam vrne naslednji izpis:

```
Check Level                Mode                    Status
=============================================================
Database connection        -connection             OK
Schema                     -schema                 OK
Datafiles consistency      -verify_db_files        OK
```

Preslikamo stolpec `Check Level` izpisa med atribute `db` značke v naši strukturi XML, kot prikazuje tabela 5.1, za vrednosti atributa `status` pa uporabimo podatke iz stolpca `Status`.

Atribut izpisa	**Atribut v XML**
Database connection	connection
Schema consistency	schema
Datafiles consistency	datafiles

Tabela 5.1: Preslikava atributov izpisa v atribute XML.

Zdravje odjemalcev preverimo tako, da domenska imena odjemalcev preberemo iz datoteke XML in za vsakega izmed njih poženemo ukaz `gdmclientcheck -client ‘‘domensko_ime’’ -check`. S tem ukazom preverimo odzivnost odjemalcev in ali je dosegljiv in aktiven. Rezultat je sledeč:

Client Host	Type	Status
Node1.gdm.com	linux	Active
Node2.gdm.com	win	Inactive
Node3.gdm.com	win	Failed
Node4.gdm.com	linux	Active
Node5.gdm.com	win	Active

Izpis prečistimo in vsakemu izmed odjemalcev v datoteki XML vpišemo v `status` ustrezno vrednost. Enak postopek uporabimo tudi za pridobitev stanja naprav. Poženemo ukaz `gdmdevicestatus -id ’’guid_naprave’’ -status`, ki nam izpiše informacije o napravah:

GUID	Type	S.M.A.R.T.	Filesystem	Status
fe362500-e033-442c-bfc5-dc907120c8f3	HDD	OK	ntfs	OK
74db138d-bbe6-4583-bd41-0621d6a99864	HDD	OK	ext4	Warning
d448ff28-fe5c-493d-8df1-361547cd533f	VHD	OK	ext4	OK
e831f883-cc84-463f-9e42-a3a4a854b83c	SSD	Warning	ntfs	Failed
831d8f86-1b00-4e85-8cb6-2338b79e6dcb	HDD	OK	ext4	OK
6983e9b3-fe22-4239-ad46-dbbc1b974631	iSCSI	OK	zfs	Warning
51816943-5007-473c-a19f-2f9902d71e23	iSCSI	OK	zfs	OK

5.4.3 Korak nadzorovanja dogodkov

Pri nadzorovanju spremljamo tudi dogodke, ki jih piše GDM v svojo dnevniško datoteko. Tu so zbrani vsi dogodki, katere ne moremo zajeti med preverjanjem zdravja objektov, vendar želimo za njih tvoriti sporočila, saj lahko prikazujejo napake v delovanju. Dnevniška datoteka vsebuje zapise v obliki, kjer vsaka vrstica ustreza enemu zapisu:

```
<severity> <timestamp> <component_name> <error_type> <
    error_message>
```

Preverimo ali je nastopil časovni interval za pričetek koraka nadzorovanja dogodkov. Iz datoteke XML preberemo atribut `last_pos` v znački `event`, ki nam označuje na kateri vrstici smo ostali pri zadnjem branju dnevniške datoteke. V kolikor atribut ne obstaja ali pa ima vrednost 0 pomeni, da moramo pričeti brati na začetku. Podatke iz prebrane vrstice ločimo z uporabo regularnih izrazov, jih preslikamo v atribute ukaza `opcmsg` ter zgeneriramo in pošljemo sporočila za vsako prebrano vrstico. Ko končamo z branjem v atribut `last_pos` vnesemo število zadnje prebrane vrstice in `last_time`, ki označuje kdaj smo nazadnje izvedli korak nadzorovanja dogodkov. Ker se dogodki beležijo v dnevniško datoteko zelo počasi, njihova obdelava pa je hitra, nam ni potrebno skrbeti, da bi med njenim branjem in obdelavo podatkov prišlo do tako velikega števila novih vnosov, da bi se znašli v neskončni zanki.

5.4.4 Korak nadzorovanja objektov

Tu preverimo ali se je stanje nadzorovanih objektov od zadnjega cikla spremenilo. Zaradi zaznavanja samo sprememb v statusih dosežemo, da uporabnik ne bo za isto napako prejemal več istih sporočil, kar bi zmanjševalo preglednost in oteževalo njegovo delo. Sprememba stanja sporoča, da je prišlo do napake ali pa je bila odpravljena. V vsakem primeru pa moramo spremembo sporočiti uporabniku. Preverjanje sprememb poteka tako, da preberemo `gdm_agent_out.xml`, kjer za objekte hranimo atributa `status` in `previous_status`. V zanki preverjamo vse `service`, `client`, `db` in `device`

objekte in če `previous_status` ne vsebuje vrednosti to nakazuje, da se je korak zagnal prvič in takrat atribut inicializiramo na `OK` vrednost. Ob naslednjem ciklu zopet za objekte preverimo ali sta `status` in `previous_status` enaka. Če sta pomeni, da do spremembe ni prišlo in lahko nadaljujemo z naslednjim objektom. Ko zaznamo razliko, preberemo atribut `status` in glede na njegovo vrednost pošljemo `opcmsg` sporočilo, ki ima nastavljeno ustrezno resnost težave. Ko smo trenutno stanje obdelali, njegovo vrednost prepišemo v atribut `previous_status`. Korak zaključimo, ko se sprehodimo čez vse objekte v datoteki XML.

5.4.5 Korak posodabljanja topologije

Topologija nadzorovanega okolja se lahko spreminja z dodajanjem strežnikov, novih odjemalcev in naprav. To lahko povzroči, da uporabnik v administratorski konzoli vidi stanje, ki ne odraža realnega stanja okolja, saj nismo izvedli posodobitve. Ta problem rešimo z implementacijo mehanizma za posodabljanje servisne hierarhije. Definirali smo tri načine posodobitve. V samodejnem načinu servisno hierarhijo posodobimo vsakič, ko zaznamo spremembo, z uporabo samodejne akcije. V drugem načinu uporabnika le obvestimo, da je do spremembe prišlo, posodobitev pa mora sprožiti ročno, za kar bomo uporabili operatersko akcijo. V zadnjem načinu pa servisno hierarhijo posodobimo na vnaprej določen časovni interval, ki ga uporabnik lahko spreminja. Podatek, s katerimi opišemo način posodabljanja hranimo v konfiguracijskem delu datoteke XML. Vrsto posodabljanja opisuje atribut `mode` značke `update`. Da zadostimo pogoju za periodično posodabljanje v atributu `interval` hranimo čas v sekundah, ki določa kdaj naj se posodobitev izvede. Funkcija, ki skrbi za posodobitev, prebere datoteko XML, ter preveri za kakšno vrsto posodobitve gre. V kolikor je vrednost nastavljena na periodično, preverimo ali ustrezamo intervalu za zagon postopka za posodabljanje. Nato z `opcmsg` pošljemo sporočilo, ki ga prestreže pravilo, ki ustreza tej vrsti sporočil. V pravilu definiramo samodejno akcijo, ki na strežniku HPOM povzroči posodobitev servisne hierarhije, z zagonom

skripta za tvorjenje topologije. V primeru samodejne in ročne posodobitve, preverjanje časovnega intervala preskočimo in pošljemo sporočilo. Ker se samodejna in periodična posodobitev med seboj razlikujeta le v tem kdaj se katera časovno izvede lahko pri samodejni uporabimo isto sporočilo. V primeru ročne posodobitve pa pošljemo drugačno sporočilo, saj mora pravilo nanj reagirati drugače. Kadar zazna, da gre za ročno posodobitev uporabnik prejme sporočilo z definirano operatersko akcijo, ki jo lahko požene.

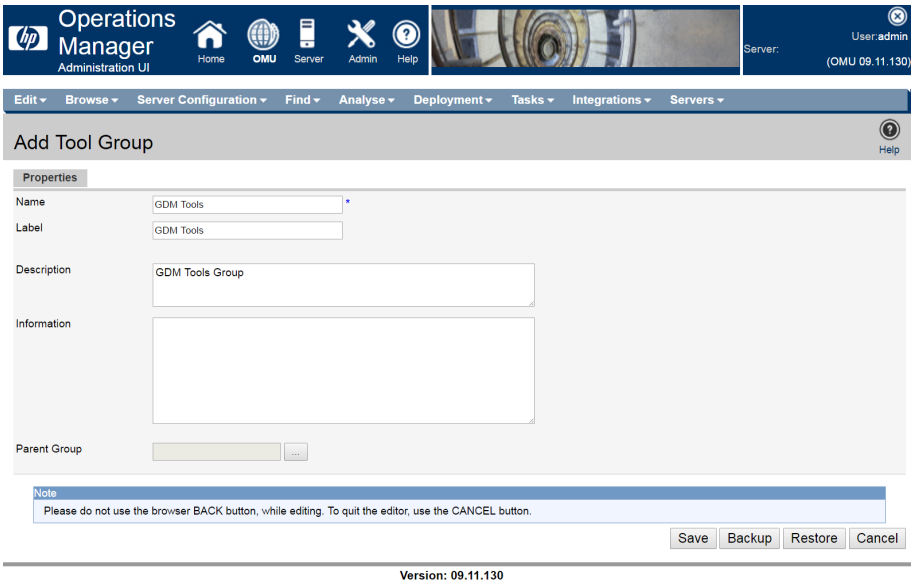
5.5 Konfiguracija HP Operations Manager

5.5.1 Aplikacije

HPOM omogoča operaterjem zaganjanje aplikacij. Aplikacije so lahko skripti, privzeta programska orodja, ki jih ponuja HPOM, integrirane aplikacije ali pa tiste, ki jih namesti vtičnik SPI. Aplikacije lahko integriramo v različne komponente HPOM:

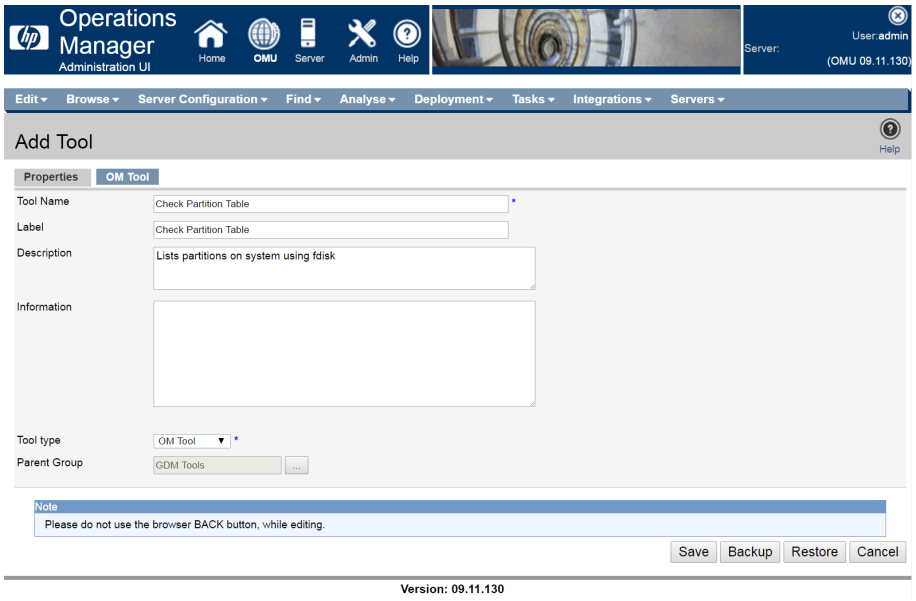
- integrirana aplikacija v administratorko konzolo,
- akcije v sporočilih,
- servisno hierarhijo.

V vtičniku SPI uporabljamo aplikacije tako v administratorski konzoli, kjer jih uporabnik lahko najde v mapi z orodji, kot tudi akcije na prispela sporočila. Prikazali smo kreiranje samodejne akcije in akcijo, ki jo izvrši operator ob prejetju določenega sporočila. Z integracijo aplikacij tako še dodatno razširimo zmožnosti nadzorovanja vtičnika SPI. Konfiguracijo in kreiranje aplikacij naredimo preko administracijske konzole v HPOM [2].
Aplikacije oziroma orodja, ki jih uporabljamo pri nadzorovanju našega produkta zaradi lažje preglednosti razporedimo v ločeno skupino. V meniju izberemo Tools Group in kliknemo Add Tool Group (slika 5.4). Vpišemo ime, oznako, ter opis skupine in pritisnemo gumb shrani.



Slika 5.4: Dodajanje skupine orodij.

Nato izberemo opcijo Add Tool in vpišemo podatke o orodju (slika 5.5). Za Tool Type izberemo OM tool in za Parent Group izberemo skupino, ki smo jo predhodno kreirali.



Slika 5.5: Lastnosti orodja.

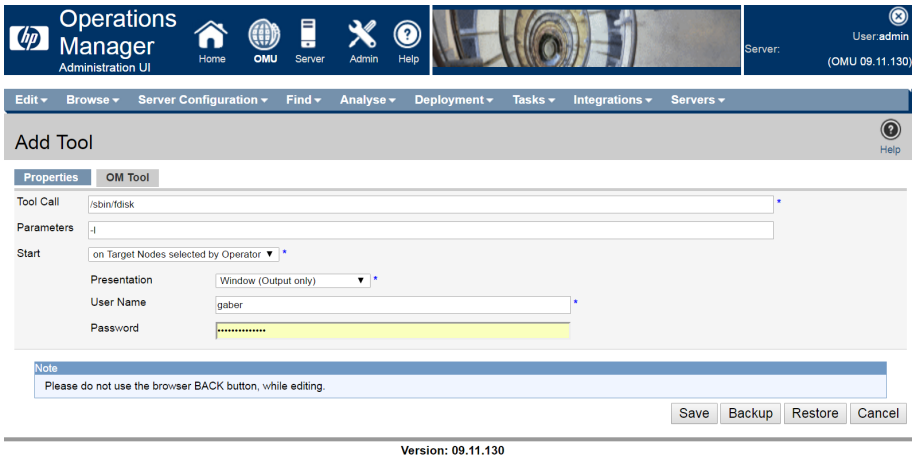
Na zavihku OM Tool (slika 5.6) vnesemo Tool Call, ki predstavlja ukaz ali skript, ki ga želimo pognati. V polje Parameters vnesemo parametre, ki jih ukaz ali skripta potrebuje za izvajanje. Nato v Start izberemo kje želimo, da se orodje izvede. Možnosti, ki nam jih HPOM ponuja so:

- upravljavski strežnik,
- vozlišče, ki ga izbere operator,
- lokalni odjemalci,
- URL naslov lokalnega spletnega brskalnika,
- seznam vozlišč, ki jih izbere operator.

Uporabnika bo zanimalo stanje na upravljanem vozlišču, zato bomo tu izbrali možnost vozlišče, ki ga izbere operator. Zaradi varnostnih mehanizmov moramo podati uporabniško ime in geslo, s katerim se bo orodje pognalo. V primeru, da uporabnik nima pravic za poganjanje orodja, bo HPOM to preprečil in vrnil sporočilo o napaki. Na tak način definiramo še ostala orodja,

ki uporabniku omogočajo dodatne funkcionalnosti pri uporabi.
Seznam orodij:

- defragmentacija diska,
- pregledovanje diska,
- izpis vrednosti S.M.A.R.T. parametrov,
- poročilo o zmogljivosti diska,
- testiranje diska pod obremenitvijo,
- kreiranje particij na disku,
- brisanje diska,
- poročilo o slabih sektorjih.



Slika 5.6: Definicija orodja.

Novo ustvarjena orodja prenesemo s klikom na gumb prenesi in jih shranimo. Definicija orodja je prikazana v dodatku A.2.

5.5.2 Pravila

Pravilo je konfiguracijski element, ki nudi mehanizem za procesiranje informacij iz različnih virov. Pravilo je predstavljeno kot ena ali več datotek XML, ki vsebuje podatke in meta informacije. Del pravila, ki vsebuje podatke po navadi sestavljajo pravila za tvorjenje sporočila na upravljanemu vozlišču, kjer je pravilo nameščeno. Podatkovni del je v celoti definiran s strani uporabnika, del, ki vsebuje meta informacije pa s strani HPOM in se uporablja v administracijske namene. Pravila ob namestitvi vtičnika SPI distribuiramo na vsa upravljana vozlišča, s katerih želimo zbrati in procesirati določene dogodke. Procesiran dogodek nato postane sporočilo, ki se pošlje na strežnik HPOM. Večina pravil obdeluje sporočila, ki jih tvorimo s pomočjo `opcmsg` in `opcmon` ukazov ali aplikacijskega programskega vmesnika, lahko pa je vir tudi dnevniška datoteka. Poznamo več vrst pravil, ki se razlikujejo po tem, kakšno stanje definirajo, kaj je njihov vir informacij ter kako preoblikujejo dogodke v sporočila. Poznamo naslednje vrste [4]:

- Open Message Interface:
  Procesirajo dogodke kreirane z ukazom `opcmsg`. Uporablja se za tvorjenje sporočil HPOM s strani obstoječih aplikacij.

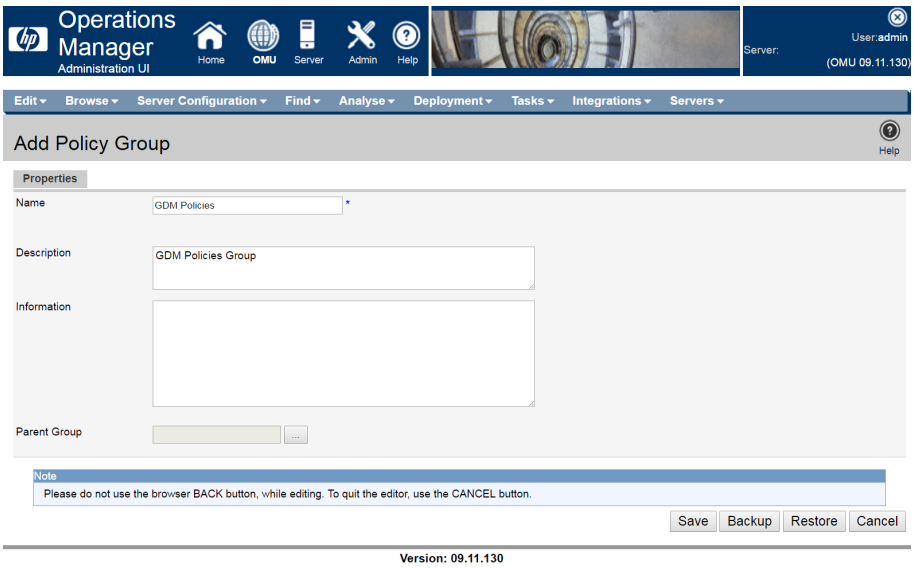
- Measurement Threshold:
  Pridobivanje podatkov se vrši s pomočjo ukaza `opcmon` kateri pošilja številske vrednosti metrik. Uporablja se predvsem za nadzorovanje zmogljivosti zunanjih aplikacij.

- Scheduled Task:
  Tip pravila, ki je primeren za ponavljajoče dogodke, saj se vrši na časovni interval.

- Logfile Encapsulation:
  Uporablja se za nadzorovanje dnevniških datotek.

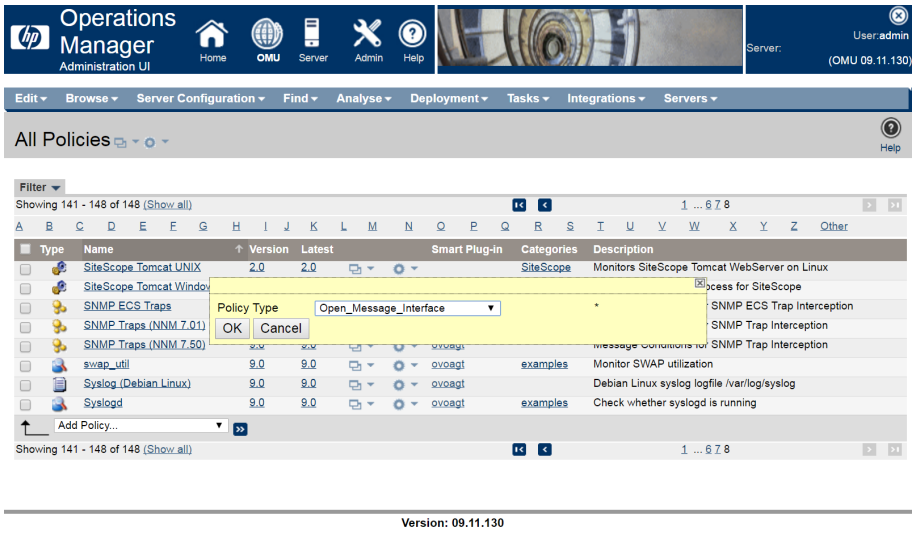
V našem vtičniku SPI uporabljamo pravila, ki prestrezajo dogodke, ki jih proži agent, ko zazna, da je prišlo do neželenega stanja. Vrsta, ki najbolje

ustreza je Open Message Interface. Pravila, ki prestrezajo določeno vrsto sporočil, razvrstimo po skupinah in tako dosežemo večjo preglednost in jih lažje distribuiramo na upravljana vozlišča. HPOM omogoča distribucijo celotne skupine pravil. V primeru, da na kakšnem upravljanem vozlišču ne želimo prejemati sporočil za določen vidik nadzorovanja pri distribuciji preprosto izpustimo tisto pravilo ali kar celotno skupino.
Kreiranje pravil naredimo skozi administracijsko konzolo na strežniku HPOM. V prvem koraku kreiramo skupino pravil, ki bo združila sorodna pravila. V administratorski konzoli izberemo pogled All policies groups in pritisnemo Add. Vpišemo ime in kratek opis in skupino kreiramo (slika 5.7).



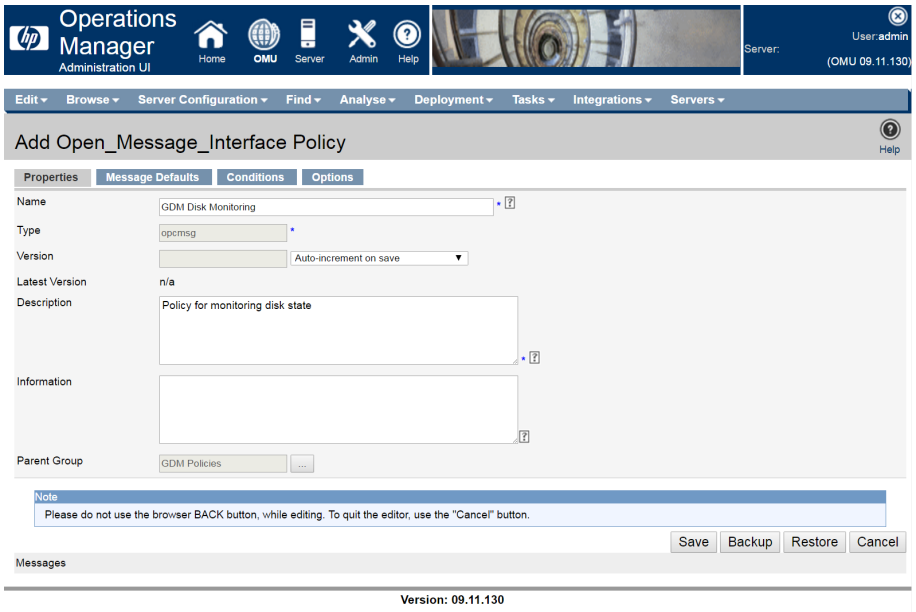
Slika 5.7: Dodajanje skupine pravil.

Zatem izberemo pogled All policies in izberemo akcijo Add policy. Izberemo vrsto Open_Message_Interface (slika 5.8).



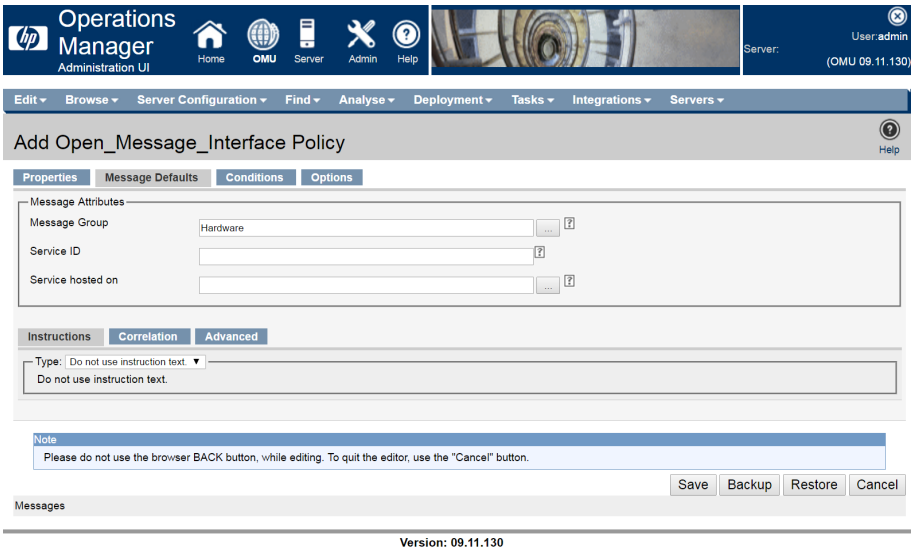
Slika 5.8: Vrsta pravila.

V Properties zavihku (slika 5.9) vpišemo zahtevane podatke in v Parent Group izberemo skupino ki smo jo ustvarili.



Slika 5.9: Lastnosti pravila.

V zavihku Message Defaults (slika 5.10) v polju Message Group izberemo skupino Hardware, kar pomeni, da bodo vsa tvorjena sporočila tega pravila spadala v to skupino. V tem zavihku lahko določimo privzete vrednosti, vendar ker želimo uporabljati različne vrednosti za različna sporočila, bomo ostale izpustili.



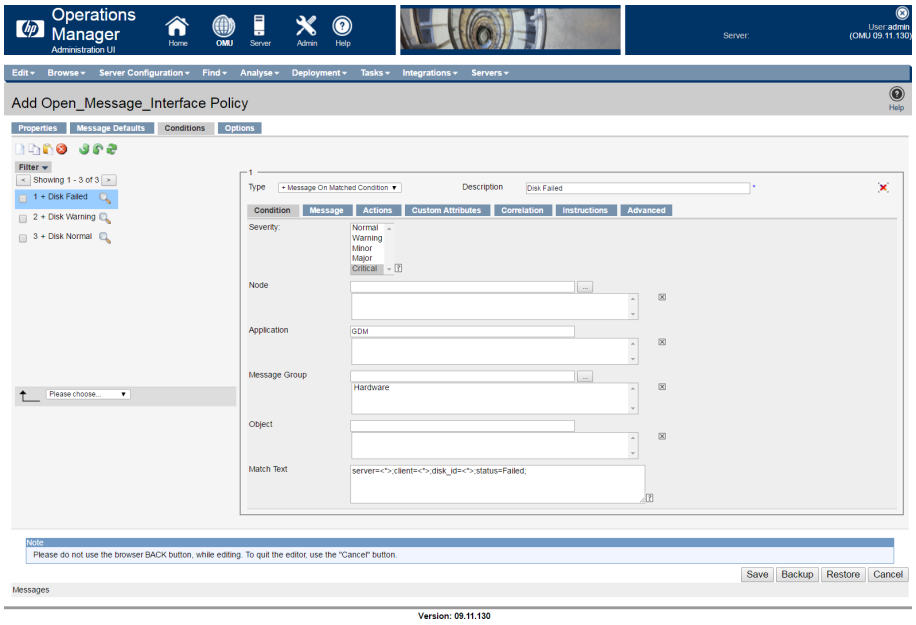
Slika 5.10: Privzete vrednosti pravila.

V naslednjem koraku dodamo različne pogoje, prikazane na sliki 5.11, s katerimi ločimo sporočila med seboj. Tako lahko za sporočila določimo različne parametre kot so:

- resnost težave,
- besedilo sporočila,
- skupino sporočil,
- vozlišče, ki je poslalo sporočilo.

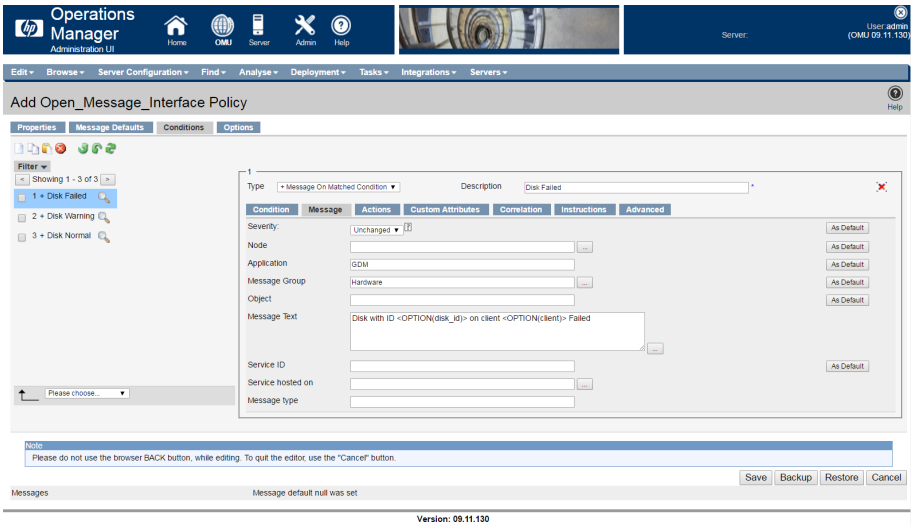
Za vsakega izmed pogojev izpolnimo stanje resnosti, ime aplikacije in ujemanje besedila. V to polje vpišemo besedilo, ki se ujema s tistim, ki smo

ga poslali z uporabo `opcmsg` ukaza. Na voljo so tudi izrazi, za primerjavo vzorcev. Tako dosežemo, da prestrežemo vsa sporočila, ki vsebujejo določen vzorec, kar je uporabno kadar vnaprej ne poznamo določenih delov sporočil.



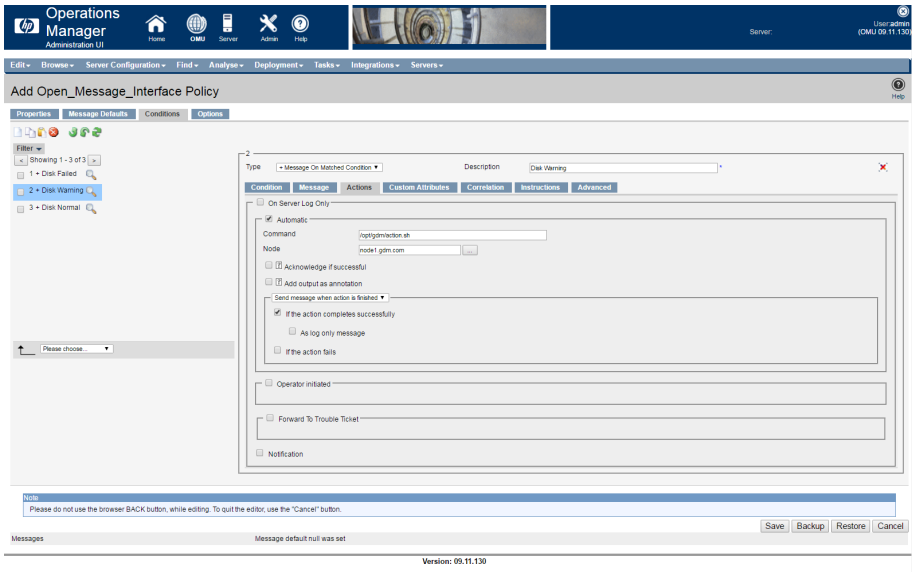
Slika 5.11: Pogoji pravila.

V zavihku Message (slika 5.12) lahko, če želimo določimo nove vrednosti za sporočilo, ki smo ga prestregli. V našem primeru spremenimo samo besedilo, da bo bolj berljivo, ostale atribute pa pustimo nespremenjene.



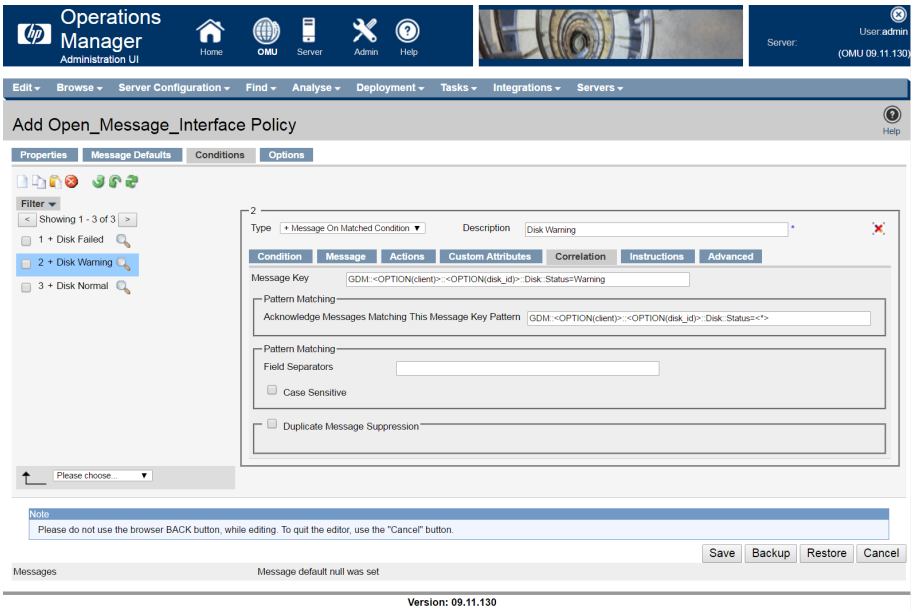
Slika 5.12: Sporočilo pravila.

Samodejno ali operatersko akcijo definiramo v zavihku Actions, kot prikazuje slika 5.13. Vpišemo pot do skripta ali ukaza, ki želimo, da se požene, določimo vozlišče na katerem naj se požene in izberemo opcijo naj se po uspešno zaključeni akciji pošlje sporočilo.



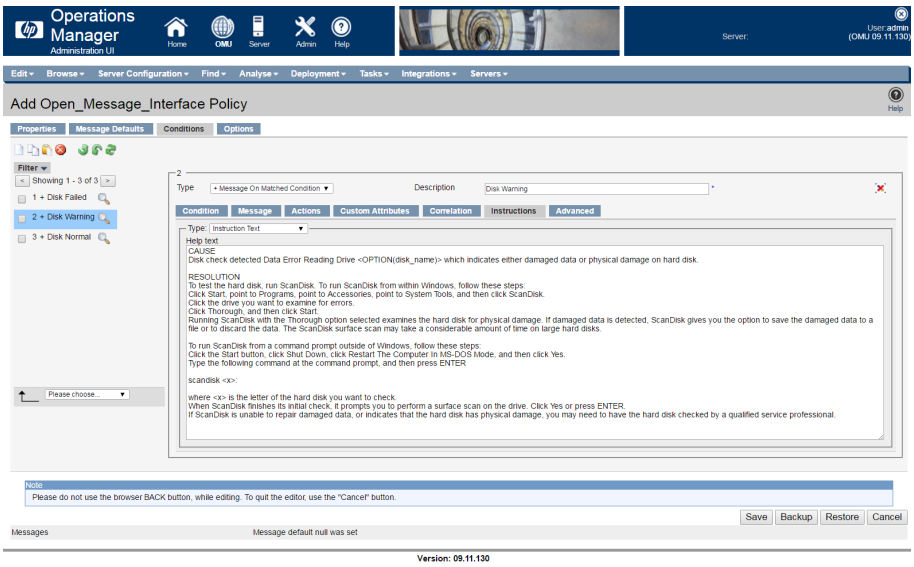
Slika 5.13: Samodejna akcija pravila.

HPOM nudi uporabniku samodejno potrjevanje sporočil ki ustrezajo določenemu ključu. Kadar pride do napake, uporabniku pošljemo sporočilo o napaki z ustreznim statusom. Uporabnik mora nato težavo rešiti in potrditi. V izogib nepreglednosti zaradi velikega števila sporočil in napakam pri njihovem potrjevanju, lahko to rešimo samodejno. Vsakemu sporočilu določimo enoličen ključ. Kasnejše prejetim sporočilom, ki se bodo nanašali na isto težavo, pa bomo povedali ključe sporočil, ki naj jih samodejno potrdi (slika 5.14).
Primer: Na sistemu je prišlo do 85 odstotne zasedenosti sistemskega diska. Uporabnik bo prejel sporočilo z opozorilom o težavi. Ko bo to odpravil bo sporočilo, ki ga bo prejel po tem, prejšnje sporočilo o napaki samodejno potrdilo, in uporabniku ne bo več vidno.
Avtomatsko potrjevanje nastavimo v zavihku Correlation, kjer določimo enoličen ključ in vzorec, po katerem bo iskal ujemanje ključa in izvedel potrjevanje.



Slika 5.14: Korelacija sporočil pravila.

V zavihku Instructions vstavimo navodila, ki bodo svetovala uporabniku kako odpraviti težavo. Ta korak je prikazan na sliki 5.15.



Slika 5.15: Definicija navodil sporočila.

Ko končamo z urejanjem pravila in dodajanjem pogojev tega prenesemo in shranimo. Končna oblika pravila je v dodatku A.3.
Postopek ponovimo in ustvarimo pravila za prestrezanje vseh sporočil, ki jih bomo pošiljali.

5.5.3 Servisna hierarhija

Nadzorovano infrastrukturo prikažemo v administratorski konzoli v pogledu Service Navigator, kot hierarhično topologijo. Uporabnik ima tako možnost grafične predstave okolja in lažji pregled nad delovanjem, saj se v primeru nepravilnosti objekti v topologiji obarvajo glede na resnost težave. Servisno hierarhijo implementiramo z konfiguracijsko datoteko XML. Glede na to, da je topologija dinamična, zaradi sprememb v infrastrukturi, jo tvorimo programsko. Ker podatke o objektih topologije že hranimo v `gdm_agent_out.xml` napišemo Perl skript, ki jo preoblikuje v servisno konfiguracijsko datoteko. Pri tem moramo zadostiti XSD shemi, ki nam jo zapoveduje HPOM [4]. Konfiguracijska datoteka servisne hierarhije se nahaja na HPOM strežniku. Prvi korak pri tvorjenju topologije je pridobitev XML datotek z zbranimi podatki z vseh upravljanih vozlišč, ki jih želimo prikazati. Njihove podatke lahko dobimo tako, da na strežniku HPOM izvedemo ukaz `opcnode -list_nodes`, ki izpiše vsa vozlišča, ki so v HPOM podatkovni bazi. Ko imamo znana imena vozlišč za katera bi radi tvorili topologijo, na strežniku poženemo ukaz `ovdeploy` kateri omogoča izmenjavo datotek med strežnikom in upravljanimi vozlišči. Prenesemo XML datoteke s podatki in jih shranimo na strežnik. Zatem vsako datoteko preberemo in njeno vsebino pretvorimo v servisno konfiguracijsko datoteko XML. Zopet uporabljamo Perl modul XML::Simple za delo s strukturami XML. Pravilnost strukture in sintakse naše izdelane konfiguracijske datoteke lahko preverimo z uporabo ukaza `opcservice -check`. Nato aktiviramo servisno konfiguracijo z ukazom `opcservice -add` in jo dodelimo ustreznim skupinam uporabnikom, kar storimo z `opcservice -assign ''operator'' ''ime_servisa''`.
Prikaz datoteka z izdelano topologijo se nahaja v dodatku A.4.

5.5.4 Registracija komponente v HP Operations Manager

Sedaj ko smo naredili konfiguracijo HPOM ter končali z implementacijo agenta za zbiranje podatkov, bomo le-tega registrirali kot proces , ki ga bo samodejno upravljal HPOM [6]. komponenta HPOM za upravljanje s procesi
-
t ovcd, skrbi za zagon, ustavljanje, ponovno zaganjanje in za pravilen vrstni red izvajanja vseh procesov HPOM. Procesi se samodejno registrirajo v `ovcd` komponento in jih lahko upravljamo preko ukaza ukazne vrstice `ovc` z opcijami `-start`, `-stop` in `-status`. Vsak proces ima svojo registracijsko datoteko XML, ki opisuje kako se s procesom upravlja. Ker ločimo procese na procese upravljavskega strežnika in procese upravljanega vozlišča, se njihove konfiguracijske datoteke, nahajajo v ustreznem direktoriju ali na upravljavskem strežniku ali upravljanemu vozlišču. HPOM uporabniku omogoča registracijo tudi lastnih procesov v `ovcd` komponento, zato si oglejmo postopek, kako to dosežemo. Proces, ki ga želimo registrirati v HPOM je naš agent, kateri želimo da teče samodejno in ga upravlja HPOM.
Za registracijski XML našega procesa bomo uporabili vzorec, ki je priložen HPOM in ga priredili našim potrebam. Opis atributov [6]:

- Name - Ime naše komponente, ki mora biti unikatno
  - `GDM_SPI`
- Label - labela komponente, ki bo vidno ko izpišemo status komponente
  - `GDM Smart Plugin`
- Description - opis komponente
  - `Smart Plugin for disk management`

- Category - komponenta lahko pripada eni ali več kategorijam. S tem lahko vršimo operacije na celotni kategoriji, kar nam olajša delo.
  - `GDM`
- Options
  - AllowAttach - ali se lahko HPOM priključi na komponento brez da bi jo bilo potrebno ustaviti
    - `false`
  - AutoRestart - ali želimo, da se komponenta ponovno zažene v primeru nepričakovane zaustavitve
    - `true`
  - MentionInStatus - ali želimo da se stanje komponente vključi v poročilo o stanju
    - `true`
  - Monitored - ali želimo, da se napake beležijo, če se komponenta nepričakovano ustavi
    - `true`
  - StartAtBootTime - ali želimo, da se komponenta zažene ob zagonu HPOM
    - `true`
  - CoreProcess - ali želimo, da se komponento lahko ustavi le z opcijo `-kill`
    - `false`
  - IsContainer - ali želimo komponento obravnavati kot komponento, ki vsebuje druge komponente

    * `false`

  – AutoShutdown - ali želimo, da se krovna komponenta ustavi, če so vse njene komponente ustavljene

    * `false`

  – AutoShutDownTimer - kako dolgo po tem, ko so vse komponente ustavljene ustavimo njihovo krovno komponento

    * `30s`

  – PollingInterval - interval kako pogosto se posodablja status stanja komponente

    * `30s`

  – ProcessDescription - Vsaka komponenta mora imeti opis procesa. Vrednost `ProcessDescription` se primerja z imenom procesa, ki je del procesne tabele operacijskega sistema. Če primerjava uspe se HPOM poveže s tem procesom.

    * `gdm_agent.pl`

  – OnHook - komponenta je po navadi proces, ki ima svoj življenjski cikel in različna stanja: ustavljen, v zagonu, v inicializaciji, v teku, v ustavljanju. Za vsako od stanj lahko definiramo različne akcije. V primeru enostavne komponente kot je naša uporabimo le akcijo za zagon procesa.

    * `START`

XML registracijska datoteka z vnesenimi vrednostmi se nahaja v prilogi A.6. Sintaktično pravilnost registracijske datoteke XML lahko preverimo z uporabo ukaza HPOM `ovcreg -check gdm.xml`, ki nam izpiše morebitne napake. Registracijo komponente v HPOM opravimo z ukazom `ovcreg -add gdm.xml`. Nato poizkusimo ali je registracija uspela in vse deluje pravilno. Poženemo `ovc -stop ‘‘ime_komponente’’`, `ovc -start ‘‘ime_komponente’’`

in `ovc -status ‘‘ime_komponente’’`in spremljamo dogajanje.

```
[root@server.gdm.com opccustproc]# /opt/OV/bin/ovc -status
adminui      OMU Administration UI Server          SERVER,OPC,
   UI (7565)    Running
coda         OV Performance Core                   COREXT
         (7952)    Running
gdm_spi      GDM Smart Plugin                            GDM
         (8543)          Stopped
opcacta      OVO Action Agent                      AGENT,EA
       (8082)    Running
opcactm      OMU Action Manager                    SERVER,OPC
     (8069)    Running
opcbbcdist   OMU BBC Config Adapter                SERVER,OPC
     (7941)    Running
[root@server.gdm.com opccustproc]# /opt/OV/bin/ovc -start gdm_spi
[root@server.gdm.com opccustproc]# /opt/OV/bin/ovc -status
adminui      OMU Administration UI Server          SERVER,OPC,
   UI (7565)    Running
coda         OV Performance Core                   COREXT
         (7952)    Running
gdm_spi      GDM Smart Plugin                            GDM
                (8543)    Running
opcacta      OVO Action Agent                      AGENT,EA
       (8082)    Running
opcactm      OMU Action Manager                    SERVER,OPC
     (8069)    Running
opcbbcdist   OMU BBC Config Adapter                SERVER,OPC
     (7941)    Running
```

V kolikor želimo odstraniti komponento poženemo
`ovcreg -del ‘‘ime_komponente’’`.
Z uspešno registracijo komponente, smo povezali vtičnik SPI v celovito rešitev namenjeno nadzorovanju aplikacije GDM.

5.5.5 Namestitev vtičnika Smart Plugin

Zaradi želje po preprosti uporabi in boljši uporabniški izkušnji, uporabniku ponudimo paket, ki poskrbi za samodejno namestitev. Paket vsebuje konfiguracijo pravil, aplikacij, datoteko za registracijo komponente, skript agenta za zbiranje podatkov, skript za tvorjenje topologije ter ostale podporne skripte. Naloga uporabnika je, da paket namesti na strežnik HPOM, kjer bo nadziral delovanje okolja. Naslednji korak pri namestitvi je porazdelitev pravil na ustrezna vozlišča, ki bodo pošiljala sporočila strežniku HPOM. V tem koraku, se na vozlišče prenese tudi instrumentacija, ki jo sestavlja agent za zbiranje podatkov in ostali podporni skripti. Vse te akcije lahko opravimo preko ukazne vrstice, kar nam omogoča, da proces instalacije preprosto skriptiramo in s tem uporabniku olajšamo delo. Uporabnik se mora prijaviti v administracijsko konzolo s svojim uporabniškim računom in že lahko prične z delom.

Poglavje 6

Sklepne ugotovitve

V diplomskem delu smo opisali glavne značilnosti nadzorovanja področja IT delovanja. Spoznali smo programsko orodje HPOM, njegove komponente in koncepte njegovega delovanja. V praksi se na IT področju srečujemo z različnimi aplikacijami in sistemi, ki nudijo podporne funkcije poslovanju podjetij. Ker izpad teh aplikacij predstavlja za podjetje časovno ali finančno izgubo si želimo v primeru napak te čim hitreje odpraviti ali jih celo predvideti. Administratorji se zato poslužujejo programskih rešitev za nadzorovanje preko katerih so obveščeni o delovanju sistemov. Razvijalci aplikacij za nadzorovanje ne morejo predvideti vseh področij, ki bi si jih želeli sistemski administratorji nadzorovati, lahko pa omogočijo razširjanje teh področij z možnostjo razvoja vtičnikov oziroma dodatnih programskih paketov, ki nudijo nadziranje specifičnih aplikacij ali sistemov. Uporabnik si tako lahko izdela lastno rešitev in si jo prilagodi po svojih željah in potrebah, ki so morda specifične le za njegovo okolje. Za primer smo vzeli aplikacijo, ki v okolju skrbi za upravljanje diskovnih pomnilnikov. Prikazali smo postopek razvoja vtičnika SPI od implementacije agenta, ki zbira podatke o sistemu do konfiguracije, ki skrbi za interpretacijo dogodkov in izvajanje popravljalnih akcij. Končni izdelek je programski paket z enostavno namestitvijo in uporabniku omogoča nadziranje z grafičnim prikazom topologije okolja, opisljivimi sporočili in navodili o reševanju napak.

Z znanjem razvoja vtičnika SPI in in dobrim poznavanjem IT domene si lahko podjetja razvijejo lastne rešitve za nadziranje kritično pomembnih sistemov in aplikacij. Svoje rešitve lahko ponudijo tudi drugim podjetjem za katere menijo, da se srečujejo s podobnimi težavami. Zaradi širine področja, ki ga nadziranje in obveščanje zajema, bi lahko dodatno implementirali še večje število orodij s katerimi bi izvajali dodatne akcije kot so priklic izbrisanih objektov na disku, konfiguracija RAID polj, spremembe velikosti sektorja in več pravil za obvladovanje sporočil. Vpeljali bi lahko Measurement Threshold vrsto pravil s katerimi bi merili zmogljivosti okolja. Na podlagi teh vrednosti bi lahko tvorili poročila, ki bi nam pomagala pri odločanju o upravljanju IT sistema. Pomembno je, da s povečanjem funkcionalnosti ne vplivamo na zmogljivosti delovanja sistema, ki ga nadziramo. Pogosto vprašanje strank pred namestitvijo SPI je, kakšen je njihov vpliv na sistemske vire. Uporabniki si želijo čimbolj enostavno namestitev in konfiguracijo ter enostaven način za spreminjanje pravil in posodobitev obstoječe konfiguracije na upravljanih vozliščih. Implementirali bi lahko mehanizem, ki bi preverjal verzijo konfiguracije na vozlišču in strežniku in bi v primeru spremembe sprožil samodejno posodobitev. Dodali bi lahko tudi dodatno servisno hierarhijo, ki bi prikazovala na kakšen način so naprave in odjemalci povezani med seboj. Ločili bi lahko tudi med tipi naprav. Ali gre za SATA, SCSI ali diskovna polja, je diskovna hramba v omrežju in ali imamo opravka z virtualnimi diski. Propagacijo statusov v servisni hierarhiji bi lahko nadgradili tudi s pogoji. Propagacija bi se tako izvedla le kadar bi določen delež diskov v odjemalcu bil v neustreznem stanju. Glavna izboljšava, ki bi jo lahko vpeljali pa je napovedovanje napak. Z metodologijami podatkovnega rudarjenja bi spremljali prejeta sporočila in se naučili kateri vzorci vodijo v neizogibno napako delovanja ali odpoved katere izmed komponent sistema.

Literatura

[1] Hewlett-Packard Development Company, L. P. *HP Operations Agent Concepts Guide*. Hewlett-Packard Company, 2013

[2] Hewlett-Packard Development Company, L. P. *HP Operations Manager Administration UI Help*. Hewlett-Packard Company, 2013

[3] Hewlett-Packard Development Company, L. P. *HP Operations Manager Concepts Guide*. Hewlett-Packard Company, 2013

[4] Hewlett-Packard Development Company, L. P. *HP Operations Manager Administrator's Reference*. Hewlett-Packard Company, 2013

[5] Hewlett-Packard Development Company, L. P. *HP Operations Manager HTTPS Agent Concepts and Configuration Guide*. Hewlett-Packard Company, 2013

[6] Hewlett-Packard Development Company, L. P. *HP Operations Manager Custom Process Management*. Hewlett-Packard Company, 2013

[7] S. Ligus, *Effective Monitoring and Alerting*. O'Reilly Media, 2013.

[8] A. Ma, G. Lu, D. Sawyer, EMC Corporation, S. Chandra, W. Hsu, Datrium Inc. "RAIDShield: Characterizing, Monitoring and Proactively Protecting Against Disk Failures", v zborniku Proceedings of the 13th USENIX Conference on File and Storage Technologies, Santa Clara, CA, USA, februar, 2015

[9] K. Meyler, C. Fuller, J. Joyner, A. Dominey, *System Center Operations Manager 2007 R2 Unleashed*. Pearson Education, 2010.

[10] R. L. Schwartz, B. D. Foy, T. Phoenix, *Learning Perl, Sixth Edition*. O'Reilly Media, 2011.

[11] (2014) HP Operations Manager Reference. Dostopno: http://www.softpanorama.org/Admin/HP_operations_manager/index.shtml

[12] (2016) HP Operations Manager. Dostopno: https://en.wikipedia.org/wiki/HP_Operations_Manager

[13] (2016) IBM Tivoli. Dostopno: http://www.softpanorama.org/Admin/Tivoli/ITM/index.shtml

[14] (2016) Zabbix. Dostopno: https://www.zabbix.org/wiki/Main_Page

[15] (2016) Remote procedure call. Dostopno: https://en.wikipedia.org/wiki/Remote_procedure_call

[16] (2016) Extensible Markup Language. Dostopno: https://en.wikipedia.org/wiki/xml

[17] (2016) S.M.A.R.T. Dostopno: https://en.wikipedia.org/wiki/S.M.A.R.T.

[18] (2016) Nagios. Dostopno: https://en.wikipedia.org/wiki/Nagios

[19] (2016) System Monitoring. Dostopno: https://en.wikipedia.org/wiki/System_monitoring

[20] (2016) Operations Management. Dostopno: http://joehertvik.com/operations-management/

Dodatek A

Programska koda

A.1 XML datoteka gdm_spi_data.xml po inicializaciji

```
<?xml version="1.0" encoding="utf-8"?>
<gdm>
<server id="" os="" hostname="">
  <services>
    <service name="as" displayname="GDM Application Server"
        type="winservice" status=""/>
    <service name="ms" displayname="GDM Management Server"
       type="winservice" status=""/>
    <service name="rpcss" displayname="Remote Procedure
       Call (RPC)" type="winservice" status=""/>
    <service name="dskutil" displayname="GDM Disk Utility
       Service" type="winservice" status=""/>
    <service name="defragsvc" displayname="GDM Defrag
       Service" type="winservice" status=""/>
    <service name="gdmdb" displayname="GDM Database Server"
        type="winservice" status=""/>
    <service name="storsvc" displayname="Storage Service"
       type="winservice" status=""/>
```

```
  </services>
  <db connection="" schema="" datafiles=""/>
<clients count="0">
</clients>
<devices count="0">
</devices>
</server>
<discovery time="0001-01-01T00:00:00" interval="600" valid
    ="0001-01-01T00:00:00" initial=""/>
<health time="0001-01-01T00:00:00" interval="600" valid
    ="0001-01-01T00:00:00"/>
<event time="0001-01-01T00:00:00" interval="600" valid
    ="0001-01-01T00:00:00"/>
<monitor time="0001-01-01T00:00:00" interval="600" valid
    ="0001-01-01T00:00:00"/>
<update mode="notify" time="0001-01-01T00:00:00" interval
    ="7200" valid="0001-01-01T00:00:00"/>
<log level="INFO" size="100MB"/>
</gdm>
```

A.2 Definicija orodja

```
DOWNLOAD_DATA APPLICATION
SYNTAX_VERSION 1

  APPLICATION_GROUP PSEUDO_GROUP
   SUBENTITIES APPLICATION
   {
    APPLICATION "Check Partition Table"
     SYMBOL "Software:Applic."
     TARGET 1
     LABEL "Check Partition Table"
     DESCRIPTION "Lists partitions on system using fdisk"
     APPL_CALL "/sbin/fdisk"
```

```
    INTERN_APPL_ACTION 0 ALLOW_CUSTOMIZE FALSE
    LICENSE_FLAG FALSE
    LICENSE_TEXT ""

    NODE
    {
    }
    APPL_LOGIN
    {
    }
    APPLICATION_TYPE CSM_PLATFORM_INTEGRATED
     UUID "ae950af2-63c8-71e6-0416-0a1124fb0000"
     START_IN_TERM_FLAG 2
     PARAMETERS "-l"
     USER_NAME "gaber"
     PASSWORD "59
         B959849E0406AC436486DBF84FAE763241B4BA6895765DD974
         "
};
```

A.3 Definicija pravila

```
SYNTAX_VERSION 5

OPCMSG "GDM Disk Monitoring"
        DESCRIPTION "Policy for monitoring disk state"
        MSGCONDITIONS
                DESCRIPTION "Disk Failed"
                CONDITION_ID "2ea41a46-63b8-71e6-0416-0
                    a1124fb0000"
                CONDITION
                        SEVERITY Critical
                        MSGGRP "Hardware"
                        TEXT "server=<*>;client <*>;disk_id
```

```
                =<*>;status=Failed;"
SET
        APPLICATION "GDM"
        MSGGRP "Hardware"
        MSGKEY "GDM::<OPTION(client)>::<
            OPTION(disk_id)>::DiskUsage::
            Status=Failed"
        MSGKEYRELATION ACK "GDM::<OPTION(
            client)>::<OPTION(disk_id)>::
            DiskUsage::Status=<*>" ICASE
        TEXT "Disk with ID <OPTION(disk_id)
            > on client<OPTION(client)>
            Failed"
DESCRIPTION "Disk Warning"
CONDITION_ID "2ea41c58-63b8-71e6-0416-0
    a1124fb0000"
CONDITION
        SEVERITY Warning
        MSGGRP "Hardware"
        TEXT "server=<*>;client<*>;disk_id
            =<*>;status=Warning;"
SET
        APPLICATION "GDM"
        MSGGRP "Hardware"
        MSGKEY "GDM::<OPTION(client)>::<
            OPTION(disk_id)>::DiskUsage::
            Status=Warning"
        MSGKEYRELATION ACK "GDM::<OPTION(
            client)>::<OPTION(disk_id)>::
            DiskUsage::Status=<*>" ICASE
        TEXT "Disk with ID <OPTION(disk_id)
            > on client<OPTION(client)> Not
            working correctly"
```

```
                                AUTOACTION "/opt/dp/action.sh"
                                   ACTIONNODE IP 176.74.176.186
                                   "176.74.176.186"
                                  SEND_MSG_AFTER_LOC_AA SEND_OK_MSG
                                HELPTEXT "CAUSE
Disk check detected Data Error Reading Drive <OPTION(
    disk_name)> which indicates either damaged data or
    physical damage on hard disk.

RESOLUTION
To test the hard disk, run ScanDisk. To run ScanDisk from
    within Windows, follow these steps:
Click Start, point to Programs, point to Accessories, point
     to System Tools, and then click ScanDisk.
Click the drive you want to examine for errors.
Click Thorough, and then click Start.
Running ScanDisk with the Thorough option selected examines
     the hard disk for physical damage. If damaged data is
    detected, ScanDisk gives you the option to save the
    damaged data to a file or to discard the data. The
    ScanDisk surface scan may take a considerable amount of
    time on large hard disks.

To run ScanDisk from a command prompt outside of Windows,
    follow these steps:
Click the Start button, click Shut Down, click Restart The
    Computer In MS-DOS Mode, and then click Yes.
Type the following command at the command prompt, and then
    press ENTER

scandisk <x>:

where <x> is the letter of the hard disk you want to check.
```

```
When ScanDisk finishes its initial check, it prompts you to
    perform a surface scan on the drive. Click Yes or press
    ENTER.
If ScanDisk is unable to repair damaged data, or indicates
    that the hard disk has physical damage, you may need to
    have the hard disk checked by a qualified service
    professional."
                    HELP "2ea41c76-63b8-71e6-0416-0
                        a1124fb0000"
                DESCRIPTION "Disk Normal"
                CONDITION_ID "2ea41db6-63b8-71e6-0416-0
                    a1124fb0000"
                CONDITION
                        SEVERITY Normal
                        MSGGRP "Hardware"
                        TEXT "server=<*>;client<*>;disk_id
                            =<*>;status=Normal;"
                SET
                        APPLICATION "GDM"
                        MSGGRP "Hardware"
                        MSGKEY "GDM::<OPTION(client)>::<
                            OPTION(disk_id)>::DiskUsage::
                            Status=Normal"
                        MSGKEYRELATION ACK "GDM::<OPTION(
                            client)>::<OPTION(disk_id)>::
                            DiskUsage::Status=<*>" ICASE
                        TEXT "Disk with ID <OPTION(disk_id)
                            > on client<OPTION(client)>
                            Works OK"
```

A.4 Generirana servisna hierarhija topologije okolja GDM

```
<?xml version="1.0"?>
<Services xmlns="http://www.hp.com/OV/opcsvc"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-
              instance"
          xsi:schemaLocation="http://www.hp.com/OV/opcsvc /
              etc/opt/OV/share/conf/OpC/mgmt_sv/dtds/service
              .xsd">
        <Service>
                <Name>root</Name>
                <Label>GDM</Label>
                <Icon>/opt/OV/www/htdocs/ito_op/images/hp
                    .32.gif</Icon>
                <Source>
                        <Composition/>
                        <ServiceRef>server</ServiceRef>
                </Source>
        </Service>
        <Service>
                <Name>server</Name>
                <Label>Server</Label>
                <Icon>/opt/OV/www/htdocs/ito_op/images/
                    server.32.gif</Icon>
                <Source>
                        <Composition/>
                        <ServiceRef>db</ServiceRef>
                </Source>
                <Source>
                        <Composition/>
                        <ServiceRef>services</ServiceRef>
                </Source>
```

```
        <Source>
                <Composition/>
                <ServiceRef>clients</ServiceRef>
        </Source>
</Service>
<Service>
        <Name>db</Name>
        <Label>Database</Label>
        <Icon>/opt/OV/www/htdocs/ito_op/images/
            database.gif</Icon>
</Service>
<Service>
        <Name>services</Name>
        <Label>Services</Label>
        <Icon>/opt/OV/www/htdocs/ito_op/images/
            process.32.gif</Icon>
        <Source>
                <Composition/>
                <ServiceRef>as</ServiceRef>
        </Source>
        <Source>
                <Composition/>
                <ServiceRef>ms</ServiceRef>
        </Source>
        <Source>
                <Composition/>
                <ServiceRef>rpcss</ServiceRef>
        </Source>
        <Source>
                <Composition/>
                <ServiceRef>dskutil</ServiceRef>
        </Source>
        <Source>
```

```
                <Composition/>
                <ServiceRef>defragsvc</ServiceRef>
            </Source>
            <Source>
                <Composition/>
                <ServiceRef>gdmdb</ServiceRef>
            </Source>
            <Source>
                <Composition/>
                <ServiceRef>storsvc</ServiceRef>
            </Source>
        </Service>
        <Service>
            <Name>clients</Name>
            <Label>Clients</Label>
            <Icon>/opt/OV/www/htdocs/ito_op/images/pc
                .32.gif</Icon>
            <Source>
                <Composition/>
                <ServiceRef>node1</ServiceRef>
            </Source>
            <Source>
                <Composition/>
                <ServiceRef>node2</ServiceRef>
            </Source>
            <Source>
                <Composition/>
                <ServiceRef>node3</ServiceRef>
            </Source>
            <Source>
                <Composition/>
                <ServiceRef>node4</ServiceRef>
            </Source>
```

```
            <Source>
                    <Composition/>
                    <ServiceRef>node5</ServiceRef>
            </Source>
    </Service>
    <Service>
            <Name>as</Name>
            <Label>as</Label>
            <Icon>/opt/OV/www/htdocs/ito_op/images/
                process.32.gif</Icon>
    </Service>
    <Service>
            <Name>ms</Name>
            <Label>ms</Label>
            <Icon>/opt/OV/www/htdocs/ito_op/images/
                process.32.gif</Icon>
    </Service>
    <Service>
            <Name>rpcss</Name>
            <Label>rpcss</Label>
            <Icon>/opt/OV/www/htdocs/ito_op/images/
                process.32.gif</Icon>
    </Service>
    <Service>
            <Name>dskutil</Name>
            <Label>dskutil</Label>
            <Icon>/opt/OV/www/htdocs/ito_op/images/
                process.32.gif</Icon>
    </Service>
    <Service>
            <Name>defragsvc</Name>
            <Label>defragsvc</Label>
            <Icon>/opt/OV/www/htdocs/ito_op/images/
```

```
            process.32.gif</Icon>
    </Service>
    <Service>
            <Name>gdmdb</Name>
            <Label>gdmdb</Label>
            <Icon>/opt/OV/www/htdocs/ito_op/images/
                process.32.gif</Icon>
    </Service>
    <Service>
            <Name>storsvc</Name>
            <Label>storsvc</Label>
            <Icon>/opt/OV/www/htdocs/ito_op/images/
                process.32.gif</Icon>
    </Service>
    <Service>
            <Name>node1</Name>
            <Label>node1</Label>
            <Icon>/opt/OV/www/htdocs/ito_op/images/pc
                .32.gif</Icon>
            <Source>
                    <Composition/>
                    <ServiceRef>disk0</ServiceRef>
            </Source>
            <Source>
                    <Composition/>
                    <ServiceRef>disk1</ServiceRef>
            </Source>
            <Source>
                    <Composition/>
                    <ServiceRef>disk2</ServiceRef>
            </Source>
    </Service>
    <Service>
```

```
            <Name>node2</Name>
            <Label>node2</Label>
            <Icon>/opt/OV/www/htdocs/ito_op/images/pc
                .32.gif</Icon>
            <Source>
                    <Composition/>
                    <ServiceRef>disk3</ServiceRef>
            </Source>
            <Source>
                    <Composition/>
                    <ServiceRef>disk4</ServiceRef>
            </Source>
    </Service>
    <Service>
            <Name>node3</Name>
            <Label>node3</Label>
            <Icon>/opt/OV/www/htdocs/ito_op/images/pc
                .32.gif</Icon>
            <Source>
                    <Composition/>
                    <ServiceRef>disk5</ServiceRef>
            </Source>
            <Source>
                    <Composition/>
                    <ServiceRef>disk6</ServiceRef>
            </Source>
    </Service>
    <Service>
            <Name>node4</Name>
            <Label>node4</Label>
            <Icon>/opt/OV/www/htdocs/ito_op/images/pc
                .32.gif</Icon>
            <Source>
```

```
                <Composition/>
                <ServiceRef>disk7</ServiceRef>
        </Source>
</Service>
<Service>
        <Name>node5</Name>
        <Label>node5</Label>
        <Icon>/opt/OV/www/htdocs/ito_op/images/pc
            .32.gif</Icon>
        <Source>
                <Composition/>
                <ServiceRef>disk8</ServiceRef>
        </Source>
</Service>
<Service>
        <Name>disk0</Name>
        <Label>disk0</Label>
        <Icon>/opt/OV/www/htdocs/ito_op/images/
            hdisk.32.gif</Icon>
</Service>
<Service>
        <Name>disk1</Name>
        <Label>disk1</Label>
        <Icon>/opt/OV/www/htdocs/ito_op/images/
            hdisk.32.gif</Icon>
</Service>
<Service>
        <Name>disk2</Name>
        <Label>disk2</Label>
        <Icon>/opt/OV/www/htdocs/ito_op/images/
            hdisk.32.gif</Icon>
</Service>
<Service>
```

```
            <Name>disk3</Name>
            <Label>disk3</Label>
            <Icon>/opt/OV/www/htdocs/ito_op/images/
                hdisk.32.gif</Icon>
    </Service>
    <Service>
            <Name>disk4</Name>
            <Label>disk4</Label>
            <Icon>/opt/OV/www/htdocs/ito_op/images/
                hdisk.32.gif</Icon>
    </Service>
    <Service>
            <Name>disk5</Name>
            <Label>disk5</Label>
            <Icon>/opt/OV/www/htdocs/ito_op/images/
                hdisk.32.gif</Icon>
    </Service>
    <Service>
            <Name>disk6</Name>
            <Label>disk6</Label>
            <Icon>/opt/OV/www/htdocs/ito_op/images/
                hdisk.32.gif</Icon>
    </Service>
    <Service>
            <Name>disk7</Name>
            <Label>disk7</Label>
            <Icon>/opt/OV/www/htdocs/ito_op/images/
                hdisk.32.gif</Icon>
    </Service>
    <Service>
            <Name>disk8</Name>
            <Label>disk8</Label>
            <Icon>/opt/OV/www/htdocs/ito_op/images/
```

```
                    hdisk.32.gif</Icon>
        </Service>
</Services>
```

A.5 Predloga registracije procesa XML

```
<?xml version='1.0' encoding='UTF-8' standalone='yes'?>
<ovc:OvCtrl xmlns:ovc="http://openview.hp.com/xmlns/ctrl/
    registration/1.5">
     <ovc:Component>
          <ovc:Name>ComponentName</ovc:Name>
          <ovc:Label>
               <ovc:String>ComponentLabel</ovc:String>
          </ovc:Label>
          <ovc:Category>Category</ovc:Category>
          <ovc:Options>
               <ovc:AllowAttach>false</ovc:AllowAttach>
               <ovc:AutoRestart>false</ovc:AutoRestart>
               <ovc:AutoRestartLimit>5</ovc:AutoRestartLimit>
               <ovc:AutoRestartMinRuntime>60</ovc:
                   AutoRestartMinRuntime>
               <ovc:AutoRestartDelay>5</ovc:AutoRestartDelay>
               <ovc:MentionInStatus>true</ovc:MentionInStatus>
               <ovc:Monitored>true</ovc:Monitored>
               <ovc:StartAtBootTime>true</ovc:StartAtBootTime>
               <ovc:CoreProcess>false</ovc:CoreProcess>
               <ovc:IsContainer>false</ovc:IsContainer>
               <ovc:AutoShutdown>false</ovc:AutoShutdown>
               <ovc:AutoShutdownTimer>1</ovc:AutoShutdownTimer
                   >
               <ovc:PollingInterval>30</ovc:PollingInterval>
          </ovc:Options>
          <ovc:ProcessDescription>ProcessDescription</ovc:
              ProcessDescription>
```

```
        <ovc:CommandLine>CommandLine</ovc:CommandLine>
        <ovc:OnHook>
            <ovc:Name>OnHookName</ovc:Name>
            <ovc:Actions>Actions</ovc:Actions>
        </ovc:OnHook>
        <ovc:OnEvent>
            <ovc:Name>OnEventName</ovc:Name>
            <ovc:EventOptions>EventOptions</ovc:
                EventOptions>
            <ovc:Actions>Actions</ovc:Actions>
        </ovc:OnEvent>
    </ovc:Component>
</ovc:OvCtrl>
```

A.6 Registracijski XML GDM procesa

```
<?xml version='1.0' encoding='UTF-8' standalone='yes'?>
<ovc:OvCtrl xmlns:ovc="http://openview.hp.com/xmlns/ctrl/
    registration/1.5">
  <ovc:Component>
    <ovc:Name>GDM_SPI</ovc:Name>
    <ovc:Label>
      <ovc:String>GDM Smart Plugin</ovc:String>
    </ovc:Label>
    <ovc:Category>GDM</ovc:Category>
    <ovc:Options>
      <ovc:AllowAttach>false</ovc:AllowAttach>
      <ovc:AutoRestart>true</ovc:AutoRestart>
      <ovc:MentionInStatus>true</ovc:MentionInStatus>
      <ovc:Monitored>true</ovc:Monitored>
      <ovc:StartAtBootTime>true</ovc:StartAtBootTime>
      <ovc:CoreProcess>false</ovc:CoreProcess>
      <ovc:IsContainer>false</ovc:IsContainer>
      <ovc:AutoShutdown>false</ovc:AutoShutdown>
```

```
        <ovc:AutoShutdownTimer>30</ovc:AutoShutdownTimer>
        <ovc:PollingInterval>30</ovc:PollingInterval>
      </ovc:Options>
      <ovc:ProcessDescription>gdm_agent.pl</ovc:
          ProcessDescription>
      <ovc:OnHook>
        <ovc:Name>START</ovc:Name>
        <ovc:Actions>
          <ovc:Start>
            <ovc:CommandLine>$OvBinDir/gdm/gdm_agent.pl</ovc:
                CommandLine>
          </ovc:Start>
        </ovc:Actions>
      </ovc:OnHook>
    </ovc:Component>
</ovc:OvCtrl>
```